AFRL-IF-RS-TR-2000-140
Final Technical Report
September 2000

SECURE BORDER GATEWAY PROTOCOL AND THE EXTERNAL ROUTING INTRUSION DETECTION SYSTEM

BBN Technologies

Sponsored by
Defense Advanced Research Projects Agency
DARPA Order No. G403

APPROVED FOR PUBLIC RELEASE; DISTRIBUTION UNLIMITED.

AIR FORCE RESEARCH LABORATORY
INFORMATION DIRECTORATE
ROME RESEARCH SITE
ROME, NEW YORK

AFRL-IF-RS-TR-2000-140 has been reviewed and is approved for publication.

APPROVED: [illegible]
ROBERT L. KAMINSKI
Project Engineer

FOR THE DIRECTOR: [illegible]
WARREN H. DEBANY, JR., Technical Advisor
Information Grid Division
Information Directorate

SECURE BORDER GATEWAY PROTOCOL AND THE EXTERNAL ROUTING INTRUSION DETECITON SYSTEM

Stephen T. Kent
Luis A. Sanchez

Contractor:	BBN Technologies
Contract Number:	F30602-92-C-0242
Effective Date of Contract:	30 June 1998
Contract Expiration Date:	30 December 1999
Program Code Number:	G403
Short Title of Work:	Security Enhancement and Intrusion Detection for the Border Gateway Protocol
Period of Work Covered:	Jun 98 – Dec 99
Principal Investigator:	Stephen T. Kent
Phone:	(617) 873-3988
AFRL Project Engineer:	Robert Kaminski
Phone:	(315) 330-1865

This research was supported by the Defense Advanced Research Projects Agency of the Department of Defense and was monitored by Stephen A. Scott, AFRL/IFGA, 525 Brooks Road, Rome, NY.

REPORT DOCUMENTATION PAGE	*Form Approved* *OMB No. 0704-0188*

1. AGENCY USE ONLY *(Leave blank)*	2. REPORT DATE September 2000	3. REPORT TYPE AND DATES COVERED FINAL (JUN 98 - DEC 99)

4. TITLE AND SUBTITLE SECURE BORDER GATEWAY PROTOCOL AND THE EXTERNAL ROUTING INTRUSION DETECTION SYSTEM	5. FUNDING NUMBERS C - F30602-98-C-0242 PE - 62301E PR - G403 TA - 00 WU - 01
6. AUTHOR(S) Stephen T. Kent Luis A. Sanchez	
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) BBN Technologies 10 Moulton Street Cambridge, MA 02138	8. PERFORMING ORGANIZATION REPORT NUMBER N/A
9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES) Defense Advanced Research Projects Agency 3701 North Fairfax Drive Arlington, VA 22203-1714 Air Force Research Laboratory/IFG 525 Brooks Road Rome, NY 13441-4505	10. SPONSORING/MONITORING AGENCY REPORT NUMBER AFRL-IF-RS-TR-2000-140
11. SUPPLEMENTARY NOTES AFRL Project Engineer: Robert L. Kaminski/IFGA/(315) 330-1365	
12a. DISTRIBUTION AVAILABILITY STATEMENT APPROVED FOR PUBLIC RELEASE; DISTRIBUTION UNLIMITED	12b. DISTRIBUTION CODE

13. ABSTRACT *(Maximum 200 words)*

The Border Gateway Protocol (BGP), which is used to distribute routing information between autonomous systems (ASes), is a critical component of the Internet's routing infrastructure. It is highly vulnerable to a variety of malicious attacks due to the lack of a secure means of verifying the authenticity and legitimacy of BGP control traffic. The Secure BGP projects designed a secure, scalable, deployable architecture (S-BGP) for an authorization and authentication system that addresses most of the security problems associated with BGP. This contract final report includes the following documents concerning S-BGP:

"Lessons Learned from the Secure BGP Proof-of-Concept Implementation."
"Secure Border Gateway Protocol (S-BGP)", by Stephen Kent, Charles Lynn, and Karen Seo.
"Design and Analysis of the Secure Border Gateway Protocol (S-BGP)", by Stephen Kent, Charles Lynn, and Karen Seo.

The last two items discuss the vulnerabilities and security requirements associated with BGP, describe the S-BGP countermeasures, and explain how they address these vulnerabilities and requirements. In addition, the papers provide a comparison of this architecture with other approaches that have been proposed, analysis the performance implications of the proposed countermeasures, and address operational issues.

14. SUBJECT TERMS Internet Routing, Border Gateway Protocol, Security, Public-Key Cryptography, Digital Signatures	15. NUMBER OF PAGES 100 16. PRICE CODE

17. SECURITY CLASSIFICATION OF REPORT	18. SECURITY CLASSIFICATION OF THIS PAGE	19. SECURITY CLASSIFICATION OF ABSTRACT	20. LIMITATION OF ABSTRACT
UNCLASSIFIED	UNCLASSIFIED	UNCLASSIFIED	UL

Table of Contents

INTRODUCTION

1 Secure Border Gateway Protocol

This document contains the final report for Secure Border Gateway Protocol and the External Routing Intrusion Detection System.

The Border Gateway Protocol (BGP), which is used to distribute routing information between autonomous sytems (ASes), is a critical component of the Internet's routing infrastructure. It is highly vulnerable to a variety of malicious attacks, due to the lack of a secure means of verifying the authenticity and legitimacy of BGP control traffic. The Secure BGP project designed a secure, scalable, deployable architecture (S-BGP) for an authorization and authentication system that addresses most of the security problems associated with BGP

This contract final report includes the following documents concerning S-BGP:

- "Lessons Learned from the Secure BGP Proof-of-Concept Implementation."
- "Secure Border Gateway Protocol (S-BGP)," by Stephen Kent, Charles Lynn, and Karen Seo.
- "Design and Analysis of the Secure Border Gateway Protocol (S-BGP)," by Stephen Kent, Charles Lynn, and Karen Seo.

The last two items discuss the vulnerabilities and security requirements associated with BGP, describe the S-BGP countermeasures, and explain how they address these vulnerabilities and requirements. In addition, the papers provide a comparison of this architecture with other approaches that have been proposed, analyze the performance implications of the proposed countermeasures and address operational issues.

2 External Routing Intrusion Detection System (ERIDS)

The External Routing Intrusion Detection System (ERIDS) was developed to fill a perceived lack of tools for the analysis of routing inconsistencies in the public Internet.

ERIDS captures Border Gateway Protocol (BGP) messages, analyzes them against authoritiative and observational data, and alerts network operators to inconsistencies which may reflect routing problems.

The data used in ERIDS comes from multiple sources, incorporated into two internal databases. ERIDS makes use of the Internet Routing Registries (IRRs) to form a core of authoritiative data, along with other sources such as the Internet Assigned Numbers Authority. It then supplements this data by recording observed behavior while it runs, building up information to fill in gaps in the authoritative data available. By doing so, ERIDS makes best use of the available information.

ERIDS consists of two software tools: the BGP Capture and Analysis Daemon (BCAD) and the BGP Monitoring Tool (BGP). The BCAD acts either a capture tool (server), an analysis tool (client) or both, depending on its configuration. The BGP monitoring tool subscribes to one or more BCADs' event streams and filters and displays them for later use.

The ERIDS architecture consists of BMTs subscribing to event streams from BCAD clients, each of which can subscribe to BGP message streams from BCAD servers, and listen to local BGP traffic. These servers can be emplaced at exchange points, usually on a high speed LAN or a switch port configured to provide the relevant BGP traffic.

To capture BGP traffic, the BCAD uses the libpcap library to listen to all network traffic and filter out all but BGP messages. It then discards unnecessary BGP messages, extracts the relevant fields from the BGP Update messages, compresses them and forwards them to any BCAD clients currently connected over a secure session.

The BCAD performs its analysis based on comparing the Network Level Reachability Information (NLRI), Autonomous System Path (AS Path) and Next Hop fields in the BGP Update messages against its internal databases. For each network announced in the NLRI field, the BCAD compares the source autonomous system against its databases to see if that AS is authorized to announce a route for that network, the BCAD compares each hop along the path to see if the AS's routers are configured to share routes, and the

BCAD compares the Next Hop value against known routers at the border of the final AS along the path. Each of those comparisons yields a score; if a result is found in the authoritative data, the score is either 0% or 100%; if there is no authoritative entry but there is observed data, a score in between 0% and 100% is calculated. The scores are summed, and if the summation is below a threshold value an event message is generated.

The event messages are sent over a secure session to subscribing BMTs. Each BMT can log events, present them with a graphical user interface using several filtering and sorting criteria, and perform scripted actions when events arrive.

This contract final report includes the following documents concerning ERIDS:

- "ERIDS: An External Routing Intrusion Detection System for the Internet," by Luis Sanchez. This paper introduces ERIDS and describes the architecture of the system.
- "ERIDS Design Document." This paper contains the design details of ERIDS.
- "BGP Monitoring Tool Design Document." This paper describes the BGP Monitoring Tool.
- "BCAD Design and Use Document." Describes the implementation details of the BGP Capture and Analysis Daemon or BCAD.

BBN Technologies released both BMT and BCAD source C code on August 1999.

Lessons Learned from the Secure BGP Proof-of-Concept Implementation

1. At least some Cisco routers truncate unknown path attributes to 256 bytes.

2. S-BGP should have a per-peer option to remove the Attestation path attribute when sending an UPDATE.

3. The code to verify that each prefix being advertised is covered by each RA along the path can require O(N**2) work per RA. The S-BGP code should sort the prefixes in the NLRI when generating an RA (which, recursively, translates to the first S-BGP speaker, and aggregation points) so that the work expended by the multitude of receivers is linear.

4. As mentioned above for prefixes, it may be useful to sort AS SETs. But this needs to be explored further as simple appending may be a better solution. (Appending raises the issue of what to do when an AS appears in more than one of the routes being aggregated.)

5. Some operating systems cannot dynamically grow the execution stack. Algorithms that use recursion over the RAs in a path may cause problems; looping (or tail recursion) would be a better technique. Recursion over a certification hierarchy might also be a problem.

6. The large number of public keys (10s of thousands) can be a problem for some digital signature hardware. The actual performance can be significantly less than that quoted by the vendor -- slow enough that software verification might be a better solution.

7. Some digital signature hardware cannot achieve the advertised transaction rates unless their work queue is kept full. The flow of control to keep a work queue full may require significant changes from the flow used by BGP. Keeping the work queue full might require many messages to be processed in parallel, increasing the amount of memory needed to hold the in-progress messages. A typical time-space tradeoff.

8. The information to be signed is often fragmented. Either the information needs to be marshalled together from several places, e.g., for some hardware hash implementations, or the hash function needs to be capable of accepting the information in chunks. For example., a software hash might be more efficient than one implemented in hardware. There can be a lot of pointer manipulation; using pointers might not always be the most efficient implementation technique.

Design and Analysis of the Secure Border Gateway Protocol (S-BGP)

Stephen Kent , Charles Lynn, Karen Seo
BBN Technologies

Abstract

The Border Gateway Protocol (BGP), which is used to distribute routing information between autonomous systems (ASes), is a critical component of the Internet's routing infrastructure. It is highly vulnerable to a variety of malicious attacks, due to the lack of a secure means of verifying the authenticity and legitimacy of BGP control traffic. This document describes a secure, scalable, deployable architecture, S-BGP, for a system that addresses most of the security problems associated with BGP. The paper discusses the vulnerabilities and security requirements associated with BGP, describes the S-BGP countermeasures, and explains how they address these vulnerabilities and requirements. The paper also provides a comparison of this architecture with other approaches that have been proposed, analyzes the performance implications of the proposed countermeasures, and reports on prototype implementation experience.

1 Problem Description

Internet routing is based on a distributed system composed of many routers, grouped into management domains called Autonomous Systems (ASes). Routing information is exchanged between ASes in Border Gateway Protocol (BGP) [1] UPDATE messages. BGP has proven to be highly vulnerable to a variety of attacks [2], due to the lack of a scalable means of verifying the authenticity and legitimacy of BGP control traffic. In April 1997, we began work on the security architecture described in this paper. In this section we describe the problem—how the protocol works, the nature of observed BGP traffic in the Internet, the correct operation of BGP, the threat model and BGP vulnerabilities, and the goals, constraints and assumptions that apply to the proposed countermeasures.

1.1 Overview of BGP

The BGP-4 protocol, both message syntax and the route propagation algorithm, is described in [1]. Routers implementing BGP, BGP "speakers," exchange routing information via UPDATE messages. An UPDATE message consists of three parts: a list of address prefixes for destinations that are no longer reachable via the previously specified route; a list of prefixes that are reachable; and the characteristics of the cumulative path and current inter-AS hop, contained in path attributes, that can be used to reach the address prefixes. (A prefix specifies an IP address block, and consists of a count of the initial significant bits in an IP address and the value of those bits.) The attribute used to specify the inter-AS path, the AS_PATH attribute, specifies a sequence of Autonomous Systems (ASes) along the path, each identified by its AS number. When propagating an UPDATE to a neighboring AS, the BGP speaker prepends its AS number to the sequence, and updates certain other path attributes. Since an UPDATE can specify only one path, only prefixes that share that path may be aggregated into the UPDATE.

The backbone routers of the major internet service providers (ISPs) have a route to every reachable IP address. Analysis of BGP UPDATEs recorded during January 1999, showed routing databases that contained about 61,000 IPv4 address prefixes. Each (non-leaf) BGP speaker maintains a full routing table, and sends its best route for each prefix to each neighbor speaker. When a BGP speaker reboots, it receives complete routing tables (via UPDATEs) from each of its neighbors. The worst case arises at Network Access Points (NAPs), where ISPs are connected together via a high speed (100Mb/s) LAN. A BGP speaker at a NAP might have about 30 peers.

On a daily basis, a BGP speaker at a NAP receives about 1425 UPDATEs from each peer, an average UPDATE rate of about 1 per minute per peer. This rate is affected somewhat by Internet growth, but is mostly a function of UPDATEs sent due to link, component, or congestive failures and recoveries. Analysis shows that about 50% of all UPDATEs are sent as a result of route "flaps," i.e., transient communication failures that, when remedied, result in a return to the original route. This sort of routing behavior has long been characteristic of the Internet [3] and the proposed security mechanisms take advantage of this behavior to achieve acceptable performance, as discussed in Section 6. (David Mills, an architect of the NSFNET, confirmed that route flapping has been common in the Internet since the mid-80's.)

1.2 Correct operation of BGP

Security for BGP is defined by the correct operation of BGP speakers. This definition is based on the observation that any successful attack against BGP should result in other than correct operation, presumably yielding degraded operation. Correct operation of BGP depends upon the integrity, authenticity, and timeliness of the routing information it distributes as well as each BGP speaker's processing, storing, and distribution of this information in accordance with both the BGP specification and with the (local) routing policies of the BGP speaker's AS. The following statements characterize the primary correct operation features of BGP.

- Each UPDATE received by a BGP speaker from a peer was sent by the indicated peer, was not modified en route from the peer, and contains routing information no less recent than the routing information previously received for the indicated prefixes from that peer.
- The UPDATE was intended for receipt by the peer that received it.
- The peer that sent the UPDATE was authorized to act on behalf of its AS to advertise the routing information contained within the UPDATE to BGP peers in the recipient AS.
- The owner of an address space corresponding to a reachable prefix advertised in an UPDATE was authorized by its parent organization to own that address space.
- The first AS in the route was authorized, by the owners of the address space corresponding to the set of reachable prefixes, to advertise those prefixes.
- If the UPDATE indicates a withdrawn route, then the peer withdrawing the route was a legitimate advertiser for that route, prior to its withdrawal.
- The peer that sent the UPDATE correctly applied the BGP rules and its AS's routing policies in modifying, storing, and distributing the UPDATE, in selecting the route, and in deriving forwarding information from it.
- The BGP speaker that received the UPDATE correctly applied the BGP rules and its AS's routing policies in determining whether to accept the UPDATE.

The countermeasures developed for S-BGP meet the first six of these criteria, even in the face of subversion of BGP speakers (Byzantine failures). Section 4 provides a detailed analysis of how each countermeasure contributes to correct operation. However, because the local policy features of BGP allow a speaker considerable latitude in determining how to process an UPDATE, these countermeasures cannot meet the last two criteria, i.e., such attacks could be attributed to local policies not visible outside an AS. To address such attacks, the semantics of BGP itself would have to change. Moreover, because UPDATEs do not carry sequence numbers, a BGP speaker can generate an UPDATE based on old information, e.g., withdrawing or reasserting a route based on outdated information. Thus the temporal accuracy of UPDATEs, in the face of Byzantine failures, is enforced only very coarsely by these countermeasures. (Section 5 provides more details on residual vulnerabilities.)

1.3 Threat model and BGP vulnerabilities

BGP has a number of vulnerabilities that can be exploited to cause problems such as misdelivery or non-delivery of user traffic, misuse of network resources, network congestion and packet delays, and violation of local routing policies.

Communication between BGP peers is subject to active and passive wiretapping. BGP uses TCP/IP for transport and this protocol, and its payload, can be attacked. A speaker's BGP-related software, configuration information, or routing databases may be modified or replaced illicitly via unauthorized access to a router, or to a server from which router software is downloaded, or via a spoofed distribution channel. Most of these attacks transform routers into hostile "insiders." Effective security measures must address such Byzantine attacks.

Exploitation of these vulnerabilities allows a variety of attacks. For example, fictitious BGP messages might be injected into a link (spoofing). Authentic BGP messages might be captured and either modified and re-injected into the link, combined incorrectly, or suppressed altogether. A compromised BGP speaker could generate UPDATEs for routes that do not, legitimately, pass through that speaker. All of these attacks are countered by the mechanisms described in Section 3.

UPDATE messages could be generated too frequently by a compromised BGP speaker, or the selection of routes and distribution of UPDATEs could violate the local routing policies. These failures are not addressed by the proposed countermeasures.

Better physical and procedural security for network management facilities, BGP speakers, and communication links; link-level encryption of inter-router (BGP speaker) traffic; and end-to-end encryption of management information would reduce some of these vulnerabilities. However, some aspects of such security approaches are economically unattractive or infeasible. Moreover, accidental (vs. malicious) misconfiguration would not be prevented by such measures and such misconfiguration has proven to be a source of several significant Internet outages in the past. Any security approach that leaves BGP vulnerable to such benign "attacks" violates the "principle of least privilege" and leaves the Internet routing system vulnerable at its weakest link. In contrast, the security approach described here satisfies this principle, so that any attack on any component of the routing system is limited in its impact on the Internet as a whole.

1.4 Goals, constraints, and assumptions

In order to create countermeasures that are both effective and practical, the S-BGP architecture is based on the following goals, constraints, and assumptions.

The S-BGP architecture must handle the projected growth and usage of the Internet in terms of performance (storage, processing, network bandwidth). It should be dynamic (responding automatically to topology changes, including the addition of new networks, routers and ASes) and scalable (able to handle the growth of the Internet in terms of addresses, routes, BGP control traffic, etc.).

The countermeasures must be consistent with the BGP protocol standards and with the likely evolution of these standards. This includes packet size limits, e.g., 4096 byte maximum for UPDATEs, and BGP features, e.g., path aggregation, communities, and multi-protocol support, e.g., multi-protocol label switching (MPLS). For example, to avoid modifying BGP packet formats, we have chosen to employ the BGP optional, transitive path attribute as a mechanism for distributing countermeasures information. Non S-BGP routers should pass this attribute type transparently, without understanding it.

The S-BGP architecture must be deployable. A primary goal of this work is to not only find countermeasures for BGP vulnerabilities but to cause them to be adopted by ISPs and router vendors. To accomplish this, the countermeasures must use available technology that can be incrementally deployed and they should leverage off of the existing infrastructure, e.g., the Internet Corporation for Assigned Names and Numbers (ICANN), and routing registries. In addition, they must avoid dependency loops; the architecture cannot depend on correct operation of inter-AS routing during initialization, e.g., it cannot rely on non-local databases.

2 Prior work

The earliest significant work published on the topic of routing protocol security is Perlman's doctoral dissertation [21]. The S-BGP design shares several features of that work, e.g., we address Byzantine failures and make extensive use of digital signatures. However we differ in many other respects, e.g., our design applies to a standard exterior (vs. interior) routing protocol, and we pay considerable attention to infrastructure and performance implications.

At the time that we began this work, previously published work on improving the security of BGP, and more generally distance-vector protocols, included proposals for adding sequence numbers to BGP messages [4,5,6], authentication of BGP messages [1,5,6], neighbor-to-neighbor encryption of BGP messages [4], and adding information to UPDATE messages to protect against tampering as the UPDATE propagates around the Internet [4,5,7].

None of this work proposed a comprehensive solution to the BGP security problems described above; each focused on one or more aspects of the problem without considering the full range of issues that are critical to a viable solution. For example, none addressed issues associated with the generation and distribution of public key certificates and certificate revocation lists (CRLs) needed to support validation of signed UPDATEs. Some proposals made changes to BGP that are inconsistent with the protocol standards, a reasonable approach only if one were presented with a "clean slate." None of the prior work examined the statistics of BGP operating in the Internet; this sometimes led authors to focus on performance concerns that are not the major impediment to deploying viable solutions. Some of the work developed solutions for distance vector protocols, but erroneously claimed applicability to BGP, which is described as a path vector protocol.

In contrast, the BGP security architecture reported in this paper is comprehensive, including a design for the infrastructure needed to establish and maintain the system. The optional transitive path attribute it employs is consistent with BGP standards and can be safely carried through routers not implementing S-BGP. This architecture incorporates the notion of an *address attestation*, which establishes that a "first hop" BGP speaker is authorized to advertise a route to a destination. No prior work includes an equivalent notion. Finally, the performance of the design presented here has been modeled based on actual BGP statistics, and has been tested via a live BGP feed from an ISP. No other work has been so rigorously analyzed from a performance perspective.

In [7], the scheme proposed is similar to our route attestations (RAs) in that before distributing an UPDATE to an external neighbor, the BGP speaker signs the route. Our approach differs from the scheme proposed in [7] in that we sign a routing data structure that specifies the next hop AS, explicitly indicating that this AS is authorized to advertise the route in question to the identified neighbor. Hence, our approach avoids a vulnerability not addressed in [7], i.e., provides protection against "cut and paste attacks" in which a BGP speaker inserts itself into a route (using a valid UPDATE containing a route which the speaker was not authorized to use).

The approach proposed in [4,5] was developed in response to the perceived communication and computation overhead of schemes such as [7]. In the case of [7] (and our route attestations), each of the signatures must be carried in each UPDATE and verified by each recipient to validate each received UPDATE. The approach proposed in [4,5] includes only a single signature for a route in an UPDATE; this signature covers only the destination and penultimate ASes listed in the path and is generated by a BGP speaker in the destination AS. While the overhead of [4,5] is less than that of [7] or our route attestations, it fails to provide the protection afforded by the iterated signature schemes in an environment with more sophisticated routing policies (e.g., policies other than shortest path), such as those typically supported by BGP. Moreover, our analysis shows that generation, validation, and transmission of digital signatures does not impose an unacceptable computational or communication burden (see Section 6).

3 Proposed countermeasures

The approach adopted to securing BGP route distribution involves two Public Key Infrastructures (PKIs), a new path attribute containing "attestations", and the use of IPsec. These components are used by a BGP speaker to validate the authenticity and data integrity of BGP UPDATEs that it receives and to verify the identity and authorization of the senders. This section discusses in more detail the PKIs and certificates, the attestations, the use of IPsec, and the distribution of this countermeasure information.

3.1 PKIs and certificates

S-BGP uses two PKIs, based on X.509 (version 3) certificates, to enable BGP speakers to validate the identities and authorization of BGP speakers and of owners of ASes and of portions of the IP address space. These PKIs parallel the existing IP address and AS number assignment delegation system and take advantage of this extant infrastructure. Because these PKIs mirror existing infrastructure, their creation avoids many of the "trust" issues that often complicate the creation of a PKI. The two PKIs involve four types of certificates, as illustrated below. In the diagrams:

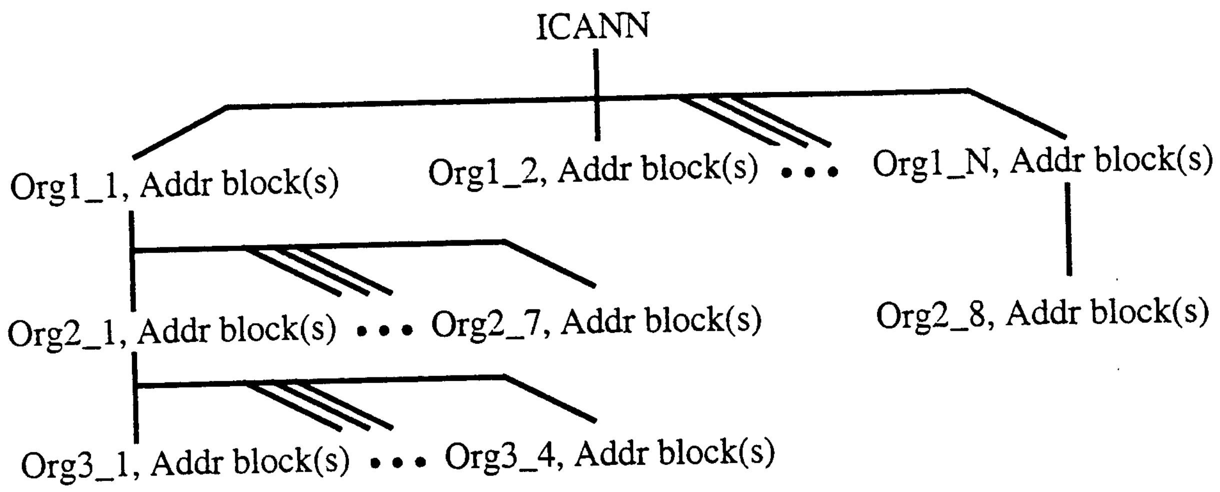
- The higher node is the issuer for the certificates defined in the tier below it.
- The name of the current tree node (organization, AS, router, etc.) is the subject of the certificate.
- Any additional fields shown in the node, e.g., address block(s), are in an extension in the certificate.
- Other X.509 certificate fields are assumed, but not shown.

Note that the organizations that assign addresses (Registries, ISPs, DSPs, etc.) and the organizations that obtain autonomous system numbers from a Registry may be different. An organization could receive its AS number from a registry and its address block from an ISP. So the Org3_4 shown in the first PKI hierarchy (Figure 1) could correspond to the Org 1 shown in the second PKI hierarchy (Figure 2).

3.1.1 A PKI for address allocation. This architecture calls for a certificate to be issued to each organization that is granted "ownership" of a portion of the IP address space. This certificate is issued through the same chain of entities that, in the existing environment, is responsible for address allocation. The root of this chain is the ICANN, followed by regional address space allocation authorities (e.g., ARIN and RIPE), ISPs, DSPs, and end users. Note that the proposed system does not require that address assignments be certified all the way to the subscriber. If a subscriber's address is allocated from that of a DSP or an ISP with which it is currently affiliated, then the certification process need only be effected as far as the ISP/DSP. The same applies to DSPs that receive their address-space assignments from ISPs. Also note that a subscriber (or a DSP) who does not participate in BGP exchanges (e.g., is singly-homed) need not be issued a certificate if the subscriber's address space is derived from that of an encompassing ISP or DSP[1]. Finally, if an organization owns multiple ranges of addresses, this design calls for assigning a single certificate[2] containing a list of address blocks, so as to minimize the number of certificates needed to validate an UPDATE.

[1] For historical reasons, the chain of issuance described above was sometimes short circuited. A subscriber (or DSP) who has an address-space assignment that has bypassed the normal allocation procedure, who has changed DSPs/ISPs and retained the originally assigned address must also be certified, even if not a BGP user.

[2] If an organization acquires additional address blocks, a new certificate is issued to reflect the increased scope of ownership.



Subject	**Description**
Org1_x	a 1st tier organization (usually a registry)
Org2_x	a 2nd tier organization (usually an ISP or DSP)
Org3_x	a 3rd tier organization (usually a DSP or user organization)

Extension	**Description**
Addr blocks(s)	one or more IP address blocks assigned to the organization

Figure 1: Address Allocation PKI Structure

This PKI reflects the assignment of address blocks to organizations by binding address block(s) to a public key belonging to the organization to which the addresses are being assigned. Unlike a typical X.509 certificate, the identity of the subject is not the primary focus of these certificates; instead, these certificates are used to prove ownership of a block of addresses[3]. Each certificate in this PKI contains a (private) extension that specifies the set of address blocks that have been allocated to the organization. The subject alternate name in each certificate is the DNS name of an organization: an ISP, DSP, or a subscriber. The ICANN, as root, is represented nominally by a self-signed certificate that contains an extension expressing ownership of the entire address space. In Figure 1:

(a) ICANN is the root and issues certificates to the first tier of organizations (Org1_x). Under current practice, Org1_x would be an Internet Registry, although historically it could have been an ISP, an organization, etc. ICANN signs the tier 1 certificates using its private key.

(b) Org1_x then assigns sub-blocks of its address space to ISPs or DSPs. In the diagram, for example, Org1_1 issues a certificate to each of Org2_1 through Org2_7 and Org1_N issues a certificate to Org2_8. Org1_x signs the certificate using the private key corresponding to the public key in the certificate it received in (a).

(c) Org2_x then assigns sub-blocks of its address space to customers, DSPs, etc. In the diagram, Org2_1 issues a certificate to each of Org3_1 through Org3_4. Org2_x signs the certificate using the private key corresponding to the public key in the certificate it received in (b).

(d) And so on....

Table 1 summarizes the Issuer/Subject relationships for the certificates in this PKI.

[3]One could place the address block in an X.509 attribute certificate, linked to an X.509 public key certificate, in a more elegant approach to representing this data. However, because the number of certificates involved is so great, and because attribute certificates are not yet widely supported, we have chosen to add the address block information as a private, v3 extension to a public key certificate.

Certificate Type	Issuer	Subject
Root	ICANN	ICANN
Registry	ICANN	Registry
ISP/DSP	Registry (or ICANN or ISP)	ISP/DSP
Subscriber	ISP/DSP (or Registry or ICANN)	Subscribe r

Table 1: Address Allocation PKI Certificate Overview

3.1.2 A PKI for ASes and router IDs. Three types of certificates will be used to support the authentication of ASes and BGP speakers, and the relationship between speakers and ASes. Here too the ICANN is the root and the next tier consists of registries, but the third tier consists of organizations that own ASes, followed by a tier of AS numbers and routers. The result is a broader, shallower certification tree. As before, this tree parallels existing "trust relationships," i.e., the ICANN assigns AS numbers to registries, which in turn assign one or more AS numbers to organizations (e.g., ISPs/DSPs) that run BGP. Each of these organizations is authoritative for identifying routers as representatives (BGP speakers) for the AS(es) that the organization owns. In order to express the ownership of an AS by an organization, each third tier certificate carries an extension that enumerates the ASes assigned to that organization. Validation of fourth tier certificates requires matching asserted AS numbers against these extensions. For each fourth tier AS certificate (b, below) there are typically several router (BGP speaker) certificates (c, below), each specifying the same AS number. (Note that there could be more than one certificate assigned to a BGP speaker if the speaker acts as a proxy for another AS.) As shown in the Figure 2, these 3 types of certificates bind together:

(a) AS numbers and an organization's public key—a registry issues these to organizations and signs them using its private key. The alternate name in the certificate is the DNS name of the organization. An extension contains the (list of ranges of) AS number(s).

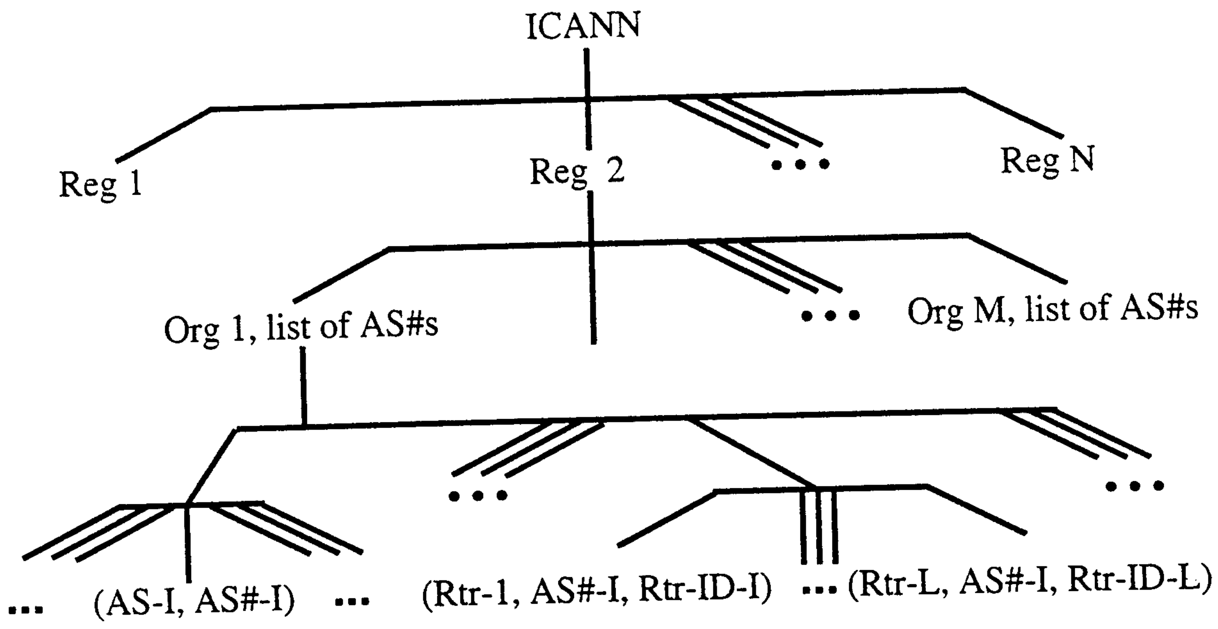
(b) an AS number and its public key—An organization issues these and signs them using the private key corresponding to the public key in the certificate described in (a). The issuer alternate name in the certificate is the DNS name of the organization. The subject alternate name is the AS number, represented as a DNS name.

(c) a router (DNS) name, a router id, an AS number, and the router's public key—An organization issues these and signs them using the private key corresponding to the public key in the certificate described in (a). Both the router id (an IP address) and the AS number are extensions in this certificate, and the binding of three items is a critical aspect of this certificate. The subject alternate name is the DNS name of the router corresponding to the router id.

3.1.3 Attestations. An attestation establishes that the subject of the attestation (an AS) is authorized by the issuer to advertise a path to the specified blocks of address space. There are two classes of attestations, address and route, although a single format is employed to represent both. Route attestations are carried in a new type of optional BGP path attribute as part of UPDATE messages:

- Address attestations—here the issuer is the organization that owns the address space and the subject is an AS that may originate it, e.g., the organization's provider. The issuer signs an address attestation using the private key that corresponds to the public key in the certificate (see Figure 1) assigning this address space to the issuer.
- Route attestations—here the subject is a transit AS. A route attestation is signed by the S-BGP speaker (or offline by the management of the AS). The signer uses the private key that corresponds to the public key in the certificate that binds the speaker to the subject AS (see Figure 2).

If an organization has more than one AS, there are separate attestations for each AS rather than just one attestation containing multiple AS numbers. Each AS will have its own set of BGP speakers and its own authentication certificate(s) as well. This applies to both the stub and transit AS cases. Figure 3 summarizes the structure for UPDATEs and route attestations.



Subject	**Description**
Reg n	DNS name of registry n
Org m	DNS name of ISP/DSP/organization m
AS-i	DNS name of autonomous system i
Rtr-j	DNS name of router j

Extension	**Description**
AS#-i	autonomous system number for AS-i
Rtr-ID-k	router identifier (IP address) for router k

Figure 2: Autonomous System Identification and BGP Speaker PKI

Certificate Type	Issuer	Subject	Extensions
Root	ICANN	ICANN	All AS #s
Registry	ICANN	Registry	AS #s
AS Owner	Registry (or ICANN)	ISP/DSP or Subscriber	AS #s
AS	ISP/DSP or Subscriber	AS number (in DNS format)	AS #
BGP Speaker	ISP/DSP or Subscriber	BGP Speaker DNS name	AS #, Router ID

Table 2: AS and BGP Speaker PKI Certificate Overview

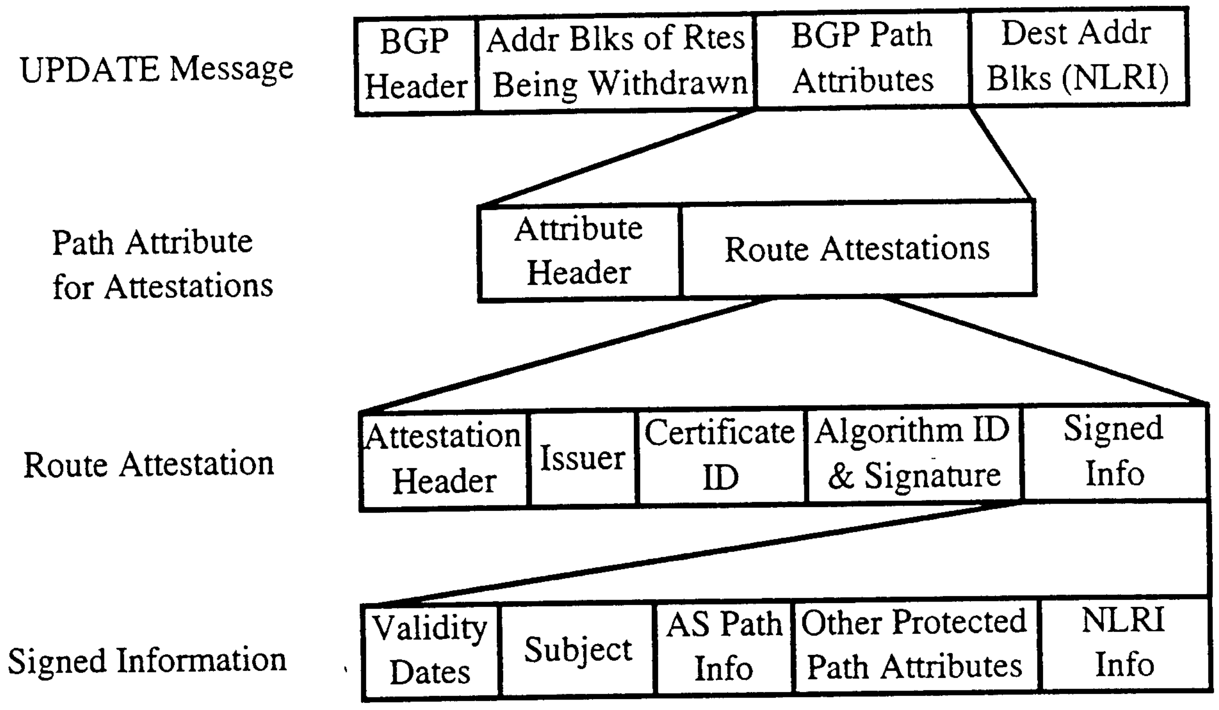


Figure 3: UPDATE Format with Route Attestations

3.2 Route validation

Attestations and certificates are used by BGP speakers to validate routes asserted in UPDATE messages, i.e., to verify that the first AS in the route has been authorized to advertise the address block(s) by the address block owner(s), and that each subsequent AS has been authorized to advertise the route for the address block(s) by the preceding AS in the route. To validate a route received from AS_n, AS_{n+1} needs:

- 1 address attestation from each organization owning an address block or blocks in the NLRI
- 1 address allocation certificate from each organization owning an address block or blocks in the NLRI
- 1 route attestation from every S-BGP speaker (or its AS) along the path (AS_n to AS_1), where the route attestation generated and signed by $router_x$ (or AS_x) specifies the NLRI and the AS_PATH from AS_{x+1} through AS_1
- 1 certificate for each S-BGP speaker along the path (AS_n to AS_1) to check the signatures on the route attestations

and, of course, all the relevant CRLs must have been verified.

This means that for each UPDATE, there must be attestations confirming that all the ASes in the BGP UPDATE are authorized to advertise routes to the destination IP address block(s). This includes ASes that are providing third party advertisements for ASes that are not running BGP.

The attestations are not used for checking withdrawn routes because the authorization of the BGP speaker to advertise those routes was verified at the time they were installed into the local routing information base (Loc-RIB). Moreover, if the BGP speaker has lost the authorization to advertise that route, then the route is by definition no longer valid and should be withdrawn.

Use of IPsec on inter-router communication paths prevents an active wiretapper from spoofing route withdrawals, or replaying valid UPDATEs at times when a BGP speaker would not transmit them, e.g., after a route has been withdrawn and prior to advertisement of the same or a different route.

3.3 Distribution of countermeasure information

This section discusses the mechanisms used to distribute certificates, CRLs, and address and route attestations to the relevant devices performing route validation: S-BGP speakers, route servers, etc.

3.3.1 Distribution of certificates, CRLs and address attestations. Each S-BGP speaker must have access to the public keys required to validate UPDATEs. For non-leaf S-BGP speakers, this amounts to a full set of certificates encompassing all address space owners, AS owners, and some number of S-BGP speakers (plus the ICANN and registry certificates). An X.509 certificate used in this environment is about 450 bytes long, depending on naming conventions and extensions. In the current (June 1999) Internet environment, there are approximately 5,300 autonomous systems, 44,000 organizations that own address prefixes, and 7,500 BGP speakers. The resulting certificate database comprises about 32 Mbytes, and it can be expected to grow each year as more address prefixes, autonomous systems, and BGP speakers are added. The CRL database associated with these certificates adds to this total, but since the certificates are issued to organizations and devices, not people, CRLs probably will not grow significantly.

At first glance it might seem appropriate to transmit certificates as part of each UPDATE. This would ensure that each receiving BGP speaker would receive all the data needed to validate the route attestations in an UPDATE, and it would be easy for each BGP speaker to include its own certificate as part of the forwarding process. However, this would be very wasteful of bandwidth, as each BGP speaker would receive many redundant copies of certificates[4]. More importantly, this approach is infeasible, because BGP UPDATEs are limited in length to 4096 bytes and thus are too small to carry the necessary certificates for most UPDATEs[5]. The introduction of a new type of BGP message for transmission of certificates (and CRLs) could address the packet size problem, but would still tend to be very wasteful of bandwidth and would not be backward compatible.

[4]The redundancy arises from several sources. A BGP speaker tends to receive routes to the same destination, via each interface, with considerable overlap of ASes in each route. Withdrawal and later re-advertisement of the same routes via the same interfaces results in additional redundancy

[5]Many UPDATEs contain routes for multiple address blocks and some routes contain many AS numbers, and each requires its own certificate.

Instead, this architecture uses out-of-band distribution of certificates and CRLs to all S-BGP speakers. This is an attractive approach to the distribution problem for several reasons. This database is relatively static and thus a good candidate for caching and incremental update. Moreover, the certificates can be validated (and processed against CRLs) and reduced to a more compact format by ISPs/DSPs prior to distribution to S-BGP speakers. (Only the public key, subject and selected extensions need be retained.) This avoids the need for each speaker to perform this processing (entailing tens of thousands of signature validations), and it saves both bandwidth and storage space. Although memory is inexpensive, most currently deployed commercial routers do not possess sufficient memory to store all of these certificates so either additional memory or auxiliary systems will be needed, even with preprocessing.

To address the distribution problem we make use of two tiers of repositories from which one can download the entire certificate and CRL database. The top tier consists of several replicated, easy to access storage sites, e.g., the NAP route servers. The second tier of repositories are operated by the ISPs/DSPs, to provide local access for the S-BGP speakers within each AS. (Per-AS repositories avoid the dependency loop that would occur if one required inter-AS routing in order to access this database.) Bulk transfer of the whole certificate or CRL database, from the first to second tier repositories, can be effected via FTP or TFTP; since certificates and CRLs are signed and carry validity interval information, there is no need for additional integrity mechanisms in the transfer. The second tier repositories also will query top tier repositories to get new CRLs, based on the NEXT UPDATE field in each CRL, and to get new certificates, based on certificate expiration, e.g., using the Lightweight Directory Access Protocol (LDAP). Retrieval of newly issued certificates (in between downloads of the complete database) for newly created ASes, organizations with new address space ownership, etc., could be effected in the same manner, from daily incremental update files.

This allows an ISP/DSP to operate in an anticipatory fashion, retrieving certificates before it needs them. Each second tier repository will validate all of the certificates and CRLs it retrieves, and produce a more compact (locally signed) database ready for consumption by the S-BGP speakers within its administrative purview. If deemed necessary, the top tier repositories also can push CRLs issued prior to scheduled dates.

The same analysis applies to address space attestations. Each address attestation requires about 100 bytes, and this amounts to approximately 4 Mbytes. Carriage of these data items in UPDATEs would usually be redundant and thus is more effectively handled via the same, out-of-band distribution mechanism. Here too, this distribution model allows pre-processing by ISP/DSP NOCs, further eliminating significant signature validation overhead (there would be roughly 44,000 such attestations currently). This model also simplifies revocation of address attestations, i.e., an address space owner can issue a CRL-like, signed data structure and include it in the database for downloading and pre-processing by each ISP/DSP. The total space required for the certificate and address attestation databases can be reduced from about 36 Mbytes to about 11 Mbytes as a result of preprocessing.

3.3.2 Distribution of route attestations. Route attestations are distributed with BGP UPDATEs in a newly defined, optional, transitive path attribute. This approach requires BGP speakers to be upgraded to a BGP release with these countermeasures. When an S-BGP speaker opens a BGP session with a peer, transmitting the advertisable portion of its routing information database via UPDATEs, relevant route attestations are sent with each UPDATE. These attestations employ a compact encoding scheme, to help ensure that they fit within the BGP packet size limits, even when route or address aggregation is employed. (We preserve S-BGP security guarantees in the face of aggregation by explicitly including signed attribute data that otherwise would be lost when aggregation occurs.) The S-BGP speaker receiving an UPDATE caches the associated attestations with the route in its routing information database. (We have not yet determined if a mechanism for revoking route attestations is required, or if a modest attestation lifetime will suffice.) Each BGP speaker generates route attestations based on receipt of UPDATEs from its neighbors, as described in Section 4.2, thus in-band distribution is appropriate. As noted below, the bandwidth required to support in-band distribution of route attestations is negligible (compared to user traffic).

3.4 IPsec and router authentication

BGP is transported over TCP and thus is protected against misordered, lost, or replayed packets, to the extent that the TCP sequence number management facility is secure. BGP-4 provides a means for carrying authentication information in BGP messages, but there is no prescribed key management scheme and there is no facility for sequence numbering of BGP messages, hence this facility is not employed here. Instead, we use the Encapsulating Security Payload (ESP) protocol (with NULL encryption), from the IP security protocol suite (IPsec) [8,9,10,11,12] to provide authentication, data integrity, and anti-replay on a point-to-point basis, i.e., between BGP speakers. (Although the Authentication Header protocol from the IPsec suite could be used here, it is less efficient and thus was not selected.) The Internet Key Exchange (IKE) protocol is used for key management services in support of ESP. The PKI established for router and AS authentication provides the necessary certificates (see Section 3.1.2).

3.5 Other issues

BGP Path Attributes are being standardized to support Communities (to support policy) [13], Confederations (used to transparently split a large AS into several smaller ASes to reduce the n^2 peering requirements) [14], and to support additional protocols (IPV6, MPLS, and multicast) [15]. The Route Attestation mechanism is designed to provide protection for these and other new path attributes, in the same manner in which it protects the current path attributes.

4 How these countermeasures address BGP vulnerabilities

This section describes how the proposed countermeasures reduce the vulnerability of BGP to the attacks described earlier and provide much of the functionality necessary for ensuring the correct operation of BGP.

4.1 Certificates

The certificates described above are used to enable verification of:

- An AS's authorization to "advertise" a block of addresses—The certificates from 3.1.1 are used to verify that an AS is authorized to "advertise" a block of addresses. Specifically, the signature on an address attestation must be verifiable using the public key in a certificate containing the address block(s) that include the address block(s) in the address attestation.
- An organization's ownership of an AS number—The certificates from 3.1.2.a are used to verify that an AS has been assigned to the holder of a particular public key, i.e., an ISP, DSP, or subscriber organization. They are used to validate 3.1.2.b or 3.1.2.a certificates through the AS number linkage.

- An AS's identity—The certificates from 3.1.2.b (or certificate data pre-processed by a NOC and distributed to routers) are used to verify the signature of an AS on a route attestation.
- A BGP speaker's identity and its association with an AS—The certificates from 3.1.2.c are used to verify the signature of a speaker on a route attestation, and in conjunction with 3.1.2.a to make sure that the speaker is authorized to act on behalf of the AS.
- Identity and authorization of a BGP peer—The certificates from 3.1.2.c are used by the BGP speakers when establishing peering sessions, to authenticate each other.

4.2 Address and route attestations

These two countermeasures support validation of the address prefixes and path information in an UPDATE.

Address attestations protect BGP against misbehaving BGP speakers that originate or distribute erroneous UPDATEs and BGP speakers whose advertisable destination addresses have been misconfigured. Whenever an S-BGP speaker advertises itself as the starting point of a route for some address prefix, other S-BGP speakers will verify that the AS represented by that speaker is the subject of an address attestation signed by the owner of the address prefix. Since only the organization that owns the prefix can sign such an attestation, no S-BGP speaker can falsify such an advertisement.

Each S-BGP UPDATE will include a set of route attestations, one per AS listed in the UPDATE, each of which is added to the UPDATE as it propagates among ASes. A route attestation indicates that the signing BGP speaker is authorized to advertise to the neighbor the route constructed thus far, by the organization owning the AS (in which the speaker resides). The route attestation is digitally signed by the S-BGP speaker distributing the UPDATE. It includes the identification of the S-BGP speaker's certificate issued by the owner of the AS, the destination addresses in the route, the list of identifiers of ASes in the route, the identifier of the AS to which the UPDATE is directed, a maximum lifetime, and other transitive data requiring protection. Each recipient of an UPDATE verifies the route attestations contained within it before deciding whether to accept and distribute the UPDATE. Route attestations protect BGP against misbehaving BGP speakers that distribute erroneous UPDATEs, and against misconfigured local routing policies.

4.3 IPsec

IPsec (specifically ESP and IKE) provides the security services needed by the receiving BGP speaker to verify message integrity, the identity of the sender, and the fact that it (the receiver) is the intended recipient of every message. Although the attestations in UPDATE messages protect against a wide range of active wiretap attacks, use of ESP provides protection for all BGP traffic, prevents replay of messages across a link, and protects TCP against various forms of attack, including SYN flooding and spoofed RSTs (resets).

5 Residual vulnerabilities

The S-BGP system (address attestations, route attestations, PKIs, and IPsec for all BGP messages) addresses many of the vulnerabilities of BGP-4. Nevertheless, there exist vulnerabilities that are not eliminated by this system, including the following:

- Suppression of BGP messages by a misbehaving BGP speaker is not addressed. Use of IPsec (and TCP) will detect active wiretap attacks that result in lost or reordered BGP packets. However, a compromised BGP speaker can elect to not transmit BGP messages, even when local policy would call for such transmission, e.g., for route withdrawal. This is undetectable by the proposed countermeasures, although coordinated network monitoring might be able to detect such misbehavior. The substantial flexibility afforded by local policies in BGP appears to preclude countering this vulnerability if such policies are to remain private to ASes (as allowed by the BGP specification and as currently practiced).
- Passive wiretapping to discover network connectivity information is not addressed. These attacks could be countered by enabling the confidentiality feature of ESP, if the risk exceeds the cost to encrypt and decrypt UPDATE messages.
- A BGP speaker may reassert a route that was withdrawn earlier, even if the route has not been re-advertised. This vulnerability exists because BGP UPDATEs do not carry sequence numbers or timestamps that could be used to determine the currency of UPDATEs. However, route attestations do expire, so there is a limit on how long an old attestation can be used for such purposes. The possibility also exists to add a CRL-like function for route attestation revocation, a possibility that will be explored later in our work.

- Verification that the BGP peers that exchanged the UPDATE, correctly applied BGP rules, local policies, etc., is not addressed. As above, BGP affords speakers considerable latitude with regard to local policy and ASes do not usually make public their local routing policies, hence it appears difficult to counter such problems. S-BGP restricts malicious behavior to the set of actions for which the speaker (or AS) is authorized, based on externally verifiable constraints.

6 Performance and operational issues

In developing the S-BGP architecture, we have paid close attention to the performance and operational impact of the proposed countermeasures. Previous work in the area of routing security has often focused almost exclusively on the costs of generating and validating digital signatures. While such costs are an important factor, our analysis suggests that the bandwidth and storage requirements associated with signatures, certificates and CRLs are much bigger problems. The following analysis is based on examination of actual Internet routing data plus simulation of the effects of S-BGP countermeasures.

6.1 Processing

The computation burden for signature generation and validation appears to be tractable in this proposed architecture. We have selected the Digital Signature Algorithm (DSA) to minimize the size of the signatures, specifically for route attestations. DSA yields only a 40-byte signature, vs. the 128-byte signature typical for RSA (using 1024-bit keys). DSA also allows for pre-computation, which permits lower latency in signature generation by S-BGP speakers. Other (S-BGP-specific) techniques (described below) significantly reduce the need for signature validation operations, so the processing asymmetry exhibited by DSA is not a concern here.

The rate at which new UPDATEs are created is not so great that signature generation and validation of route attestations is expected to pose a bottleneck. A BGP speaker at a NAP, peering with about 30 other BGP speakers, receives an average total of about .5 UPDATEs per second. Each route contains an average of 3.6 ASes, and there is one route attestation per AS, yielding a rate of about 1.8 signature validations per second. In contrast, each UPDATE generated by an S-BGP speaker requires just one signature (added by the speaker), thus the signature generation rate is less than one third (1/3.6) of this value. Peak load figures may be about a factor of ten greater, yielding a peak load of about 18 signatures per second. However, analysis of data from NAPs shows that 50% or more of all UPDATEs repeat routes already known to a BGP speaker (e.g., due to link flapping). Thus caching just one route for each address prefix enables a speaker to avoid the need to validate signatures on the vast majority of UPDATEs, reducing the peak load to about 9 signatures per second, a tolerably low computational burden[6].

Upon initialization (reboot), a BGP speaker receives complete routing table updates from each peer. This means that an S-BGP speaker will receive a very large number of route attestations requiring validation, i.e., about 220,000 per peer, in a short time interval. This sort of initialization transient would be unacceptable, even though reboots and installation of new speakers is relatively infrequent[7]. To avoid this problem, we propose the addition of non-volatile storage for validated route attestations, to preserve the cache across reboots. When installing a new BGP speaker in an AS, a NOC could dump the cache from another speaker in the AS and reload it into the new BGP speaker, to seed the cache in the newly installed device.

6.2 Transmission bandwidth

The transmission of countermeasures data in UPDATEs increases the size of these messages to approximately 450 bytes (based on an average of 3.6 route attestations per path), for a typical UPDATE that previously required only 63 bytes. This represents a significant percentage increase in BGP overhead (over 700%), but the transmission of UPDATEs represents a very, very small amount of data relative to subscriber traffic. As noted above, even a speaker at a NAP sees an average rate of less than one UPDATE per second, and such speakers are connected via 100 Mb/s interfaces today, with plans to transition to multi-Gb/s interfaces in the future.

Downloading the certificate, CRL, and address attestation databases contributes an insignificant increment to this overhead. Full database transmission, from a top tier to a second tier repository entails about a 36 Mbyte file transfer for each ISP/DSP. Even if performed on a daily basis, this traffic is swamped by subscriber file transfers. Transfers from a second tier

[6] For example, SSLeay software can perform about 40 DSA (1024-bit key) signatures per second on a 450MHz Pentium II processor.

[7] Because of the important service they provide, BGP speakers are usually afforded UPS protection and new software is deployed only after extensive testing, to minimize the likelihood of crashes.

repository to each BGP speaker in its AS are smaller, about 11 Mbytes (due to certificate extraction). Here too, even if performed daily, this would be but a drop in the ocean of subscriber traffic, and use of incremental transfers is a more likely scenario. So the impact on utilization of Internet bandwidth due to transmission of all of the countermeasures data is minimal. Also, the speed of inter-router circuits continues to increase substantially, further minimizing the impact of transmission of additional control traffic.

6.3 Storage/memory

UPDATEs received from neighbors are held by a BGP router in Routing Information Bases (RIBs) and used to generate new UPDATEs for transmission to other BGP routers. The additional memory required for pre-processed certificates and address attestations amounts to about 11 Mbytes. The space required for route attestations is about 26 Mbytes per peer, a modest amount for a typical speaker with 2 or 3 peers, but a significant amount of storage for speakers at NAPs, where each speaker has about 30 peers. This amounts to a large but feasible amount of additional RAM by current standards, where a high end workstation can be configured with hundreds of megabytes of RAM. The routers that act as BGP speakers at NAPs are large, very expensive devices. However the storage capacity of the routers currently used by ISPs/DSPs would not permit storage of S-BGP UPDATEs in their RIBs, if S-BGP were deployed. Thus additional, non-volatile storage is needed in BGP speakers to support these databases. It may be feasible to significantly reduce the storage required here, since the routes (and thus route attestations) received from different peers tend to exhibit significant overlap in their suffixes.

6.4 Deployment and transition issues

Deploying S-BGP raises a number of other issues:

Adoption of S-BGP by several groups— The ISPs, DSPs, and subscriber organizations running BGP will need to cooperate in the generation and distribution of attestations. The first tier ISPs (those connected to the NAPs) must implement the S-BGP security mechanisms in order to offer significant benefit to the Internet community. (Lower level ISPs, DSPs, and subscriber organizations will need to implement the S-BGP security mechanisms only if the expense can be justified.) ICANN and registration authorities will need to expand their operational procedures to support generation of address space and AS number delegation certificates. Finally, router vendors need to provide additional storage in next generation products, or offer ancillary devices for use with existing router products, and revise BGP software to support S-BGP.

S-BGP interaction with other exterior and interior routing protocols—External routes received from external peers need to be redistributed within the AS in order to maintain a consistent and stable view of the exterior routes across the AS. Interior routing protocols will not propagate S-BGP attestations, but if each border S-BGP speaker maintains an iBGP (The acronym "iBGP" denotes intra-AS use of BGP, in contrast to the common inter-AS use of BGP, sometimes referred to as eBGP.) connection with all other transit and border routers within the AS, this problem will be averted.

BGP-4 to S-BGP Transition—The route attestation path attribute is optional for both external and internal BGP exchanges. This allows extensive regression testing before deploying S-BGP on production equipment.

7 Test scenario and results

To verify that the proposed countermeasures work as expected, and to validate our performance projections, we created a prototype implementation of S-BGP and conducted some tests in the DARPA-sponsored CAIRN testbed.

7.1 Testbed characteristics

The testbed routers were configured with FreeBSD 2.2.x as the OS and GateD as the BGP implementation. The hardware platform consisted of Pentium Pro (200-MHz 686-class CPU) with 64 to 256 Mbytes of memory.

Our S-BGP countermeasures were added to the GateD BGP implementation. This software was instrumented to collect performance statistics as described below. We also developed a man-in-the-middle (MITM) tool that was placed between a testbed BGP speaker and a BGP peer in an ISP. This tool was used to intercept BGP control traffic coming from the BGP peer in the Internet, augment it with S-BGP countermeasure data, and then pass it to the S-BGP peer in the CAIRN testbed. In this fashion, we simulated the effect of having deployed S-BGP in the Internet. This tool also can be used to test the effectiveness and robustness of the S-BGP countermeasures, e.g., by injecting bad routes, making the link appear to go up and down, etc. Finally, we developed a tool that accepts a recording of multiple sessions of BGP traffic and replays it to simulate the original order and arrival intervals for the BGP traffic.

This latter tool allowed us to conduct tests with deterministic data, to facilitate comparison of results for BGP-4 vs. S-BGP and for optimizations of S-BGP.

For these tests, we used a short duration (10-20 minute) BGP feed, and extrapolated the results to daily operation scenarios. The feed represents a link over which about 2,900 KEEPALIVEs (55 Kbytes) and 1,426 UPDATEs (89 Mbytes) would be transmitted on a daily basis. There were about 58,000 entries in the certificate database (32 Mbytes) and about 44,000 address attestations (4 Mbytes) supporting validation of the route attestations received in these UPDATEs.

7.2 Experiment results

We tested the performance impact of the S-BGP countermeasures, vs. regular BGP, and examined the effect of some optimizations. Where results are a function of the number of peers seen by a BGP speaker, we present them on a per peering session basis. This allows one to estimate the overhead seen by a router with N peers by simply multiplying by N. Also, this provides a good approximation of how a multi-peer speaker behaves, since verification need be performed only for the "best" route. Under typical conditions, a BGP speaker receives one copy of a given UPDATE from every one of its external peers, plus one from every other BGP router in the same AS. In the absence of "best route only" validation, every one of the UPDATEs received via eBGP must be validated, whereas those received via iBGP are assumed to have been validated by the sender.

For example, at a NAP, a router might have approximately 30 eBGP peers. So one would multiply the storage and CPU numbers (which are sensitive to the number of peers) for one peer by 30 plus the number of BGP peers in the router's own AS to obtain the total RA storage and CPU needed. Additional bandwidth would be on a per link basis, so the bandwidth numbers discussed below would not have to be multiplied by the number of peers. Since validation of a route is done only for the route that is selected, the CPU costs approach those of the single peer case. (The router may not receive the "best" route first and hence may have to verify more than one UPDATE, depending on the order of arrival of different paths to the same destination, when the best one arrives, how long one waits before deciding one has the "best" route, etc.)

Bandwidth utilization is not an issue. The bandwidth used to download certificate and Address Attestation (AA) extracts, an operation that might be performed on a daily basis between a NOC and each router in its AS, is tiny compared to user traffic volumes. (The full certificate and AA database used in our experiments amounted to about 343 Mbytes, but was reduced to just 38 Mbytes as a result of the extraction pre-processing.) Even the added overhead for transmission of RAs amounted to only about 1.4Kbs between a pair of routers, again a miniscule amount relative to subscriber traffic and relative to the speed of a typical inter-router link.

Storage utilization was significant, and grew when we employed caching. The certificate extracts occupied about 32 Mbytes of storage, and AAs required about 10 Mybtes. Both figures are higher than the raw database sizes, due to the addition of data structure overhead for efficient access. This AA figure was much higher, due to storage format conventions used in GateD and in S-BGP. The AA storage figure could probably be reduced to less than 5 Mbytes with further attention to data structure design. The most significant overhead is associated with RA storage. Each interface (ADJ-RIB) required almost 17 Mbytes of additional storage, which could become significant for a router at a NAP with over 30 neighbors. When a depth one cache was added (to reduce RA validation and generation costs), the number grew to about 26 Mbytes, a 50% increase. This percentage increase is due to the fact that BGP already requires a depth one cache for inbound UPDATE data, but not for outbound data. Thus a deeper cache would result in larger percentage increases, as both inbound and outbound storage would grow.

The amount of CPU time devoted to RA validation was 123 minutes per day, while generation time was only 16.2 minutes per day. This asymmetry, roughly a factor of 8, arises because of several factors. All inbound UPDATEs were validated in the unoptimtzed case and that requires checking signatures on an average of about 3 RAs per UPDATE. Also, the DSA exhibits an asymmetry of its own, in which signature validation is slower than signature generation. Finally, not every inbound UPDATE resulted in a corresponding outbound UPDATE.

When caching was enabled, the number was largely unchanged, decreasing by only about 2%! We estimate that this is due to two factors. First, the BGP feed we received was from an ISP, and was analogous to what a DSP would receive. This means that most of the route flaps that caching would detect (and use to avoid redundant signature validation) were already filtered out by the ISP. Second, the duration of each test was relatively short, thus minimizing the opportunity for the cache to be filled and then exploited. Thus this test did not verify our thesis regarding the utility of caching. Still, the amount of time devoted to cryptographic processing was sufficiently small as to be tolerable, even using software.

8 Subsequent related work

Since we began this work, there have been several other efforts towards securing BGP. These include an assessment of BGP vulnerabilities and several approaches to addressing some of these vulnerabilities. (S. Murphy's "BGP Security Analysis" surveys this topic. At the time this paper was prepared, the most recent version was <draft-murphy-bgp-secr-03.txt>, June 1999.)

- TCP/MD5 [16]–This defines a TCP option for carrying an MD5 digest. This mechanism offers data origin authentication and data integrity, on a point-to-point basis. It protects the TCP connection used to transport BGP traffic from spoofing attacks and connection hijacking. Lack of an automated key distribution protocol complicates management and encourages overly long term use of symmetric keys. Moreover, because it fails to protect against any attacks that subvert routers or the management of routers, its overall security efficacy is quite limited.
- NLRI Origin Verification [17] – This mechanism proposes adding an address prefix delegation tree to Secure DNS [18]. For each prefix that has been authorized for use, a new resource record specifies the number of the AS that is authorized to originate that prefix. This mechanism does not address route authorization, nor does the proposal describe in detail how this data would be distributed to BGP speakers. It does represent an alternative format and database option for the address attestations developed in our architecture.
- Routing Policy System Security [19] – This approach places authorization information into a small number of databases (Internet Routing Registries) accessible by routers. The information is placed into a database by the organization responsible for the authorization, using some secure access method, e.g., SSL. It proposes that BGP speakers compare the routes that they receive to the routes listed in the database, rejecting any routes not found. No changes to BGP are required. This approach does not appear to be very dynamic, although details of how, and how often, the registries are to be accessed are omitted. Use of such registries would require ISPs/DSPs to publicize what is now local policy information, which most have refused to do. Moreover, the routes stored in the registries are not signed, so attacks against these databases, including malicious or benign errors by an ISP/DSP, could compromise security.
- Hash Chain Signatures [20] – This work describes two protocols, COSP and IOSP, based on hash chain signatures, that offer very rapid signature generation and validation in a routing protocol context. COSP is not applicable because it requires signing messages at fixed time intervals, whereas BGP generates UPDATEs on demand, as a result of topology changes; IOSP is not applicable because its efficiency depends on each router receiving essentially all routing updates, which is not characteristic of the operation of BGP.

9 Future work

Deploying this technology into the Internet will require the creation of the supporting PKIs, rooted at the ICANN, and convincing ICANN, the Internet Registries, the major ISPs, the owners of IP address blocks and the router vendors of the benefits and supportability of these security mechanisms. We hope to be able to pursue a variety of technology transfer activities to facilitate adoption of S-BGP. The prototype implementation and experiments described above are a first step in this direction.

10 Summary

BGP is a critical component of the Internet's routing infrastructure and highly vulnerable to a variety of attacks. The S-BGP countermeasures use IPsec, Public Key Infrastructure (PKI) technology, and a new BGP path attribute ("attestations") to ensure the authenticity and integrity of BGP communication on a point-to-point basis, and to validate BGP routing UPDATEs on a source to (multicast) destination basis. These enhancements will allow Internet Service Providers (ISPs) and their customers to verify that:

- reachability information they receive is from an authentic and authorized BGP peering relationship and has not been modified without authorization;
- the authorization of an organization to claim ownership of a block of IP addresses (a (sub)network) is substantiated by a chain of authorizations rooted at the Internet Corporation for Assigned Names and Numbers;
- an originating AS is authorized to advertise reachability to a block of IP addresses by the organization owning that address block;

- the ASes that processed the routing information en-route from the originating AS are not, either through mis-configuration, internal error, or compromise, advertising reachability information that is inconsistent with nominal topology;
- each AS, and its BGP speakers, that advertise a given route are identifiable and authorized to participate in global Internet routing, by a chain of authorizations rooted at the ICANN.

The significance of this work extends to essentially all Internet users, as almost all depend on the availability of global Internet for access to a wide range of services. A large and growing number of federal government services, including most DoD activities, rely heavily on the Internet for a wide range of (non-tactical) communications, hence this work has significance for national security.

11 Acknowledgements

Many individuals contributed to the design and development of S-BGP. Initial funding was provided by NSA, in April of 1997, yielding a first cut design. DARPA provided later support, under Dr. Hilarie Orman, enabling us to refine, implement and test the design. S-BGP has benefited significantly from the insight and efforts of Martha Steenstrup and Luis Sanchez. As members of the architecture team, their contributions were critical to the design of the attestation and PKI schemes and the associated evaluations of other approaches and performance and operational issues. The authors would also like to thank Michelle Casagni, for her work on the performance analysis and Joanne Mikkelson, Dennis Rockwell, and Nicholas Shectman for their efforts during the ongoing implementation and experimentation phase.

12 References

[1] Y. Rekhter, T. Li, "A Border Gateway Protocol 4 (BGP-4)," RFC 1771, March 1995.

[2] BBN Report 8217, "An Architecture for BGP Countermeasures," November 1997.

[3] C. Villamizar, R. Chandra, R. Govindan., "BGP Route Flap Damping," RFC 2439, November 1998.

[4] Smith, B.R. and Garcia-Luna-Aceves, J.J., "Securing the Border Gateway Routing Protocol," Proceedings of Global Internet '96, November 1996.

[5] Smith, B.R. Murphy, S., and Garcia-Luna-Aceves, J.J., "Securing Distance-Vector Routing Protocols," Symposium on Network and Distributed System Security, February 1997.

[6] Kumar, B., "Integration of Security in Network Routing Protocols," ACM SIGSAC Review, vol.11, no.2, Spring 1993.

[7] Murphy, S., panel presentation (no text) on "Security Architecture for the Internet Infrastructure," Symposium on Network and Distributed System Security, April 1995.

[8] S. Kent, R. Atkinson, "Security Architecture for the Internet Protocol," RFC 2401, November 1998.

[9] R. Glenn & S. Kent, "The NULL Encryption Algorithm and its Use with IPsec," RFC 2410, November 1998.

[10] S. Kent, R. Atkinson, "IP Encapsulating Security Payload (ESP)," RFC 2406, November 1998.

[11] D. Maughan, M. Schertler, M. Schneider, J. Turner, "Internet Security Association and Key Management Protocol (ISAKMP)," RFC 2408, November 1998..

[12] D. Harkins, D. Carrel, "The Internet Key Exchange (IKE)," RFC 2406, November 1998.

[13] R. Chandra, P. Traina, T. Li, "BGP Communities Attribute," RFC 1997, August 1996.

[14] P. Traina, "Autonomous System Confederations for BGP," RFC 1965, June 1996.

[15] T. Bates, R. Chandra, D. Katz, Y. Rekhter, "Multiprotocol Extensions for BGP-4," RFC 2283, February 1998.

[16] A. Heffernan, "Protection of BGP Sessions via the TCP MD5 Signature Option," RFC 2385, August 1998.

[17] T. Bates, R. Bush, T. Li, Y. Rekhter, "DNS-based NLRI origin AS verification in BGP," presentation at NANOG 12, February 1998, http://www.nanog.org/mtg-9802.

[18] D. Eastlake, 3rd, C. Kaufman, "Domain Name System Security Extensions," RFC 2065, January 1997.

[19] C. Alaettinoglu, T. Bates, E. Gerich, D. Karrenberg, D. Meyer, M. Terpstra, C. Villamizar, "Routing Policy Specification Language (RPSL)," RFC 2280, January 1998.

[20] K. Zhang, "Efficient Protocols for Signing Routing Messages," Network and Distributed System Security Symposium, March 1998.

[21] R. Perlman, ""Network Layer Protocols With Byzantine Robustness," MIT/LCS/TR-429, October, 1988.

ERIDS: An External Routing Intrusion Detection System for the Internet

Luis Sanchez
BBN Technologies
lsanchez@bbn.com

Contents

Audience

The purpose of this document is to present the architecture of an external routing intrusion detection system. This document assumes that reader has basic TCP/IP knowledge. The document further assumes that the reader has basic understanding of Internet Routing principles, the Broder Gateway Protocol and basic understanding of probability theory.

1 Introduction

1.1 Problem Statement

The Border Gateway Protocol 4 (BGP-4) is an external gateway protocol designed for distributing external routing information among routers (BGP speakers) of distinct Autonomous Systems (AS) throughout an internetwork [Rekhter95]. The integrity of the external routing infrastructure and correct operation of routers depend on the correctness of the Routing Information Base (RIB) from which the forwarding information was derived and the correct operation of the router in deriving this forwarding information. Specifically, the correct operation of BGP-4 depends upon the integrity, authenticity, and recentness of the routing information it distributes as well as each BGP speaker's processing, storing, and distribution of this information in accordance with both the BGP specification and the routing policies of the BGP speaker's autonomous system.

BGP-4 is highly vulnerable to a variety of malicious attacks both in theory and in practice, due to the lack of a scaleable means to verify the authenticity and legitimacy of BGP control traffic from which routers generate and update their Routing Information Bases. For example, the communication links between BGP neighbors are subject to active and passive wiretapping. Fictitious BGP messages could be injected into a link (spoofing attack). Intercepted BGP messages could be replayed over the link at a later time (replay attack). Regardless of the specifics of the attack, the acceptance of erroneous UPDATEs and/or suppression of UPDATEs may cause selection of routes that are not consistent with the current state of the internetwork. Compromise of management or configuration information could result in the generation of UPDATEs containing addresses of destinations outside of the autonomous system or in the selection of routes and distribution of UPDATEs in violation of the local routing policies. Attacks resulting from erroneous configuration of BGP speakers may produce the same problems as those resulting from erroneous UPDATEs produced by active wiretapping or misbehaving BGP speakers.

BBN Technologies is currently developing a system of countermeasures to secure BGP-4 [BBN97]. The system, S-BGP, relies on a Public-Key infrastructure that mimics the IP address space allocation (and autonomous system number assignment) and a new BGP transitive path attribute containing signatures to verify the validity of the routes that BGP speakers advertise. The system is backwards compatible with existing BGP-4 implementations and does not impose hardware modifications on existing routing gear. However, deployment of the proposed countermeasures is a non-trivial task. It requires deployment of additional hardware and software by Internet Service Providers (ISP) in support of these countermeasures and establishment and administration of a public key infrastructure by registration authorities and ISPs, all on the scale of the worldwide Internet. Hence it may be several years before the deployment can be completed. To mitigate this problem we decided to explore detection as a possible solution to these problems while a set of countermeasures that can prevent the above mentioned attacks are widely deployed an accepted in the Internet. ERIDS is such a solution.

In the remaining of this document we discuss ERIDS in detail starting with a system overview in Section 2. ERIDS is a rule-based system that detect routing anomalies by comparing the contents routing UPDATES against a database of Internet wide routing policies. In doing so, ERIDS uses a database structure that resembles the BGP routing information base. We discussed this structure in Section 4. In Section 5 we discuss the processing a BGP packet undergoes from when it is collected by an ERIDS probe, through the intrusion detection analysis to the event management and message display. The volume of data transfer from probes to the analysis nodes, the accuracy of detection, memory and CPU processing are among the several performance issues we discussed in Section 6. We conclude with Section 7 where we discuss several system level security issues.

2 System Overview

2.1 Concept

The ERIDS is an Intrusion Detection System specifically designed to detect routing anomalies. It is capable of: 1) collecting BGP-4 messages; 2) analyzing the messages by decoding the data payload; 3) comparing the messages against a routing information databased derived from the routing policy database(s); and, 4) detecting specific intrusion events.

The system is capable of detecting exterior routing inconsistencies by comparing actual BGP-4 UPDATEs with a routing policy database. This database contains both authoritative and non-authoritative routing policies derived from the Internet Routing Registry and history data collected at the NAPs and other access points.

The Internet Routing Registry is a set of databases distributed over the Internet which contain routing information for each Autonomous System. Networks and providers worldwide enter data directly into these registries in order to publish the set of routes originated by an Autonomous System and the routing policies implemented by it. THe IRR data however is incomplete. Due the voluntary nature of the IRR effort, not all organizations register their routes in the databases making these databases inconsistent and out of date. We discussed this in detail in Section 4

The External Routing Intrusion Detection System has a set of monitoring probes that collects BGP-4 messages at both private and public peering points and transmits them to an intrusion detection engine for decoding, analysis and response. The proposed system is capable of providing Network Operation Centers (NOCs) with information indicating incorrect operation of BGP-4 due to malicious attacks or misconfigurations. The system is capable of transmitting alert messages to NOCs upon detection of any of the following conditions:

- Unauthorized advertisement of routes
- Arrival of UPDATE messages from unidentified neighbors
- Unauthorized withdrawal of legitimate routes

All communication between the intrusion detection system and the NOC will be authenticated and encrypted (if necessary) using emerging IPSec standards. The system will be capable of verifying routing information received from both external and internal BGP peers against existing route objects in route registries and local databases. This provides the system with the means to discern database mis-configuration from malicious attacks. However, this scheme imposes further requirements on the security of the databases storing such information.

3 System Components

ERIDS is an intrusion detection system for BGP. It is comprised of two modules:

- the BGP Capture and Analysis Daemon (BCAD) and,
- the BGP Monitoring Tool (BMT).

The BGP Capture and Analysis Daemon is the heart and brain of the system. It captures BGP traces and analyzes the packets found in the traces to identify attacks or misconfiguration. It compares the packet traces against a local database of know BGP routes. Depending on the results of the analysis this module may generate alert messages with different priority levels depending on the severity of the problem.

The BGP Monitoring Tool is the client portion of the system. Then BMT serves as the event manager and message interpretation facility for the user. Messages coming from BCADs are decoded and translated into human readable format using a local event interpreter database. This tool allows the user to establish a connection to one or more BCADs in order to receive intrusion warning or alert messages from public Network Access Points and from private peering sessions simultaneously.

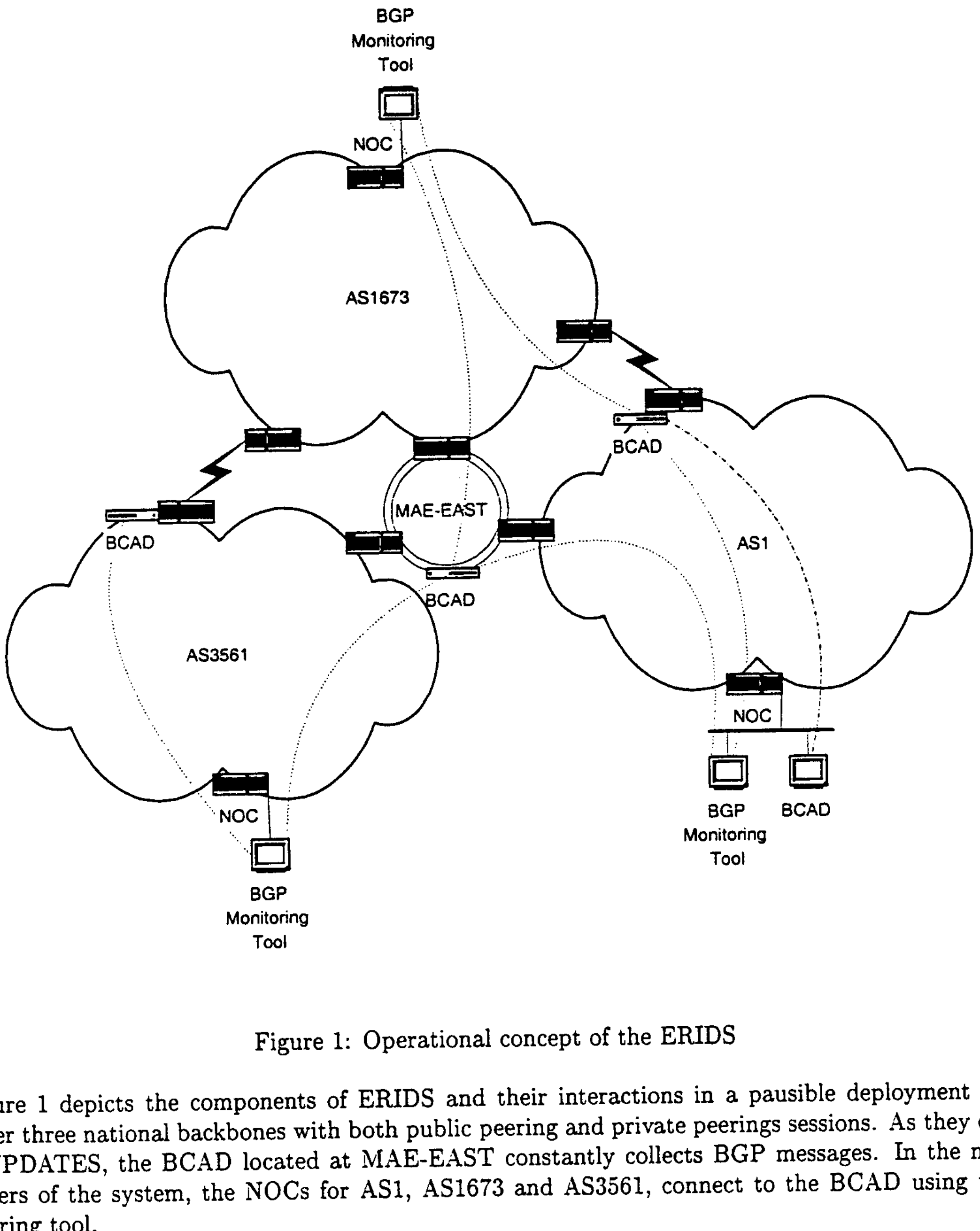


Figure 1: Operational concept of the ERIDS

Figure 1 depicts the components of ERIDS and their interactions in a pausible deployment scenario. Consider three national backbones with both public peering and private peerings sessions. As they exchange BGP UPDATES, the BCAD located at MAE-EAST constantly collects BGP messages. In the meantime customers of the system, the NOCs for AS1, AS1673 and AS3561, connect to the BCAD using the BGP Monitoring tool.

WHile collecting BGP messages the BCAD also analyzes the contents of BGP messages collected against the routing policy information contained in its databse and if necessary it generates event messages. Since the BCAD may serve multiple customers, it first replicates the event messages and forwards them to each customer NOC.

Customers may want to receive the raw data collected by a BCAD at a NAP or at a private peering location. This is appropriate since customer NOCs may want to due exhautive post-mordem analysis of the data. In figure 1 we see how AS1 is operating in this fashion. It is receiving processed data containing only event messages from the NAP's BCAD while it is receiving entire IP Datagrams collected by the BCAD located at the private peering point. These datagrams are then processed by local BCADs and any events may be displayed by the motnitoring. Like the BCAD. the BMT can handle multiple data streams from multiple sources.

The communication between BCADs or between BCADs and BMTs is protected using SSH [ref]. The system uses SSH to maximize delpoyability. The architecture does not depend on the protocol suite used to protect the data stream. IPSec [ref] can be and perhaps should be used as soon as it becomes widely deployed and supported in the Internet. The use of IPSec with automatic key mangaement and certificates to proof the identities of both BCADs and BMTs is an optimal solution from a security perspective.

3.1 BGP Capture and Analysis Daemon (BCAD)

The BCAD has three modes of operation. First it can operate as a single BGP capture facility. In this mode, BCAD captures and stores BGP packets. Secondly, BCAD can operate as a Intrusion Detection Engine alone. In this mode, BCAD receives BGP packets from probes or other BCADs, analyzes them and generates event messages. BCAD may also operate as both a probe and IDE combined. For example, one machine can use BCAD to collect BGP packets and analyze them locally.

BCAD has the facility to replicate and push multiple BGP packets or event messages simultaneosly over distinct connections to BMTs. This multiplexing facility allows BCAD to accept multiple remote connections from different ERIDS customers. Customers may decide to request that either all packets or event messages are sent to their BMTs. This enables network operators to monitor from the same location several network access points and private peering sessions simultaneously.

3.1.1 Probe

The probe module collects only IP Datagrams containing BGP messages using an interface configured in promiscous mode. The pobes first verifies that the packet received is not fragmented. Fragmention occurs when the IP datagram is larger in kilobytes than the Maximum Transmission Unit (MTU) of the interface. Because of this possiblity the probes must first identify any fragmented IP datagrams and re-assemble them before continue any processing.

¿From experience, we expect that occurence of IP datagrams fragmentation will be minimal. This is the case because typically applications that use TCP such a BGP route daemon avoid fragmentation by generating TCP segments smaller than the MTU size. TCP implementations typically avoid causing IP fragmentation by setting their Maximum Segment Size (MSS) option to be less than or equal to the MTU minus the IP header size. This option is used to inform both parties in a TCP connection of the maximum TCP segment size.

Probes also deal with the possibility of TCP segment fragmentation. We expect this to occur often especially during the establisment of a peering session when large UPDATE messages are exchanged. In BGP the maximum size of a message is 4096 bytes therefore there will be circumstances where the probes will need to re-assemble TCP segments.

Once a particular BGP packet is assembled in its entirety, the probe extracts BGP message type from the BGP message header, the NLRI, AS-PATH and Next Hop information from the body of the packet. The source MAC and IP addresses are extracted from the frame and IP datagram headers, respectively. The probes combine this and other ephemeral information such as timestamps, into one message. This is done for every single BGP message of interest (OPEN, UPDATE, NOTIFICATION) exchanged between BGP peers located at the same subnet where the probe resides. The control logic in BCAD decides if the data is transmitted to it's IDE for analysis, sent over a link to one or more remote BCADs for analysis or both. This decision depends on the configuration mode of the BCAD.

3.1.2 Database

3.1.3 Analysis Engine

The Intrusion Detection Engine (IED) is the core of the system. It takes input from the Intrusion Probes and local databases to analyze the external routing information received by BGP speakers in search of anomalies. The engine searches for discrepancies between the routing information received from different peering points (NAPs or private) and the policy information stored in the databases.

The IED has two sub-components: The Analysis and the Response Module. The analysis of the BGP UPDATEs is the responsibility of the Analysis Module. This software module translates route and autonomous system objects from the databases into a format suitable for comparison with the values found in the fields of the BGP messages. Consider the following example. Assume that the AS Object in a Routing Policy Database for autonomous system 266 appears as follows:

The routing policy for AS 266 states that this autonomous system accepts only one route from AS3561, but it accepts all routes from AS237 originated in AS237. Note from Figure 2.0, that the routes which AS266 redistributes to its neighbors also vary. For instance, AS266 announces all routes to AS3561 except the routes for AS237. Also AS237 is not given routes from AS3561. Presumably, AS237 and AS3561 peer with each other and don't need to learn their respective routes from AS266.

Under such a routing policy, an Intrusion Detection Engine in AS3561 could detect whether a BGP-4 speaker in AS266 (or an impostor) is sending UPDATE messages to AS237 with routes from AS3561 and inform the Network Operations Center of the event. The Analysis Module can also execute commands such as TRACEROUTE and PING to verify the reachability to network prefixes.

The Response Module determines the appropriate action or set of actions that the system will take based on the event identified by the Analysis Module. The module performs a search in its local database for a particular event in the events file until it finds a match. The event file contains event objects defined for the system. Consider the event object defined in Figure 3.0:

The Response Module uses the pattern field provided by the Analysis Module to locate the appropriate event. The event object contains security level labels and other descriptors indicating the appropriate actions to take in case such an event occurs. The response module provides the event output to the Event Manager.

3.2 BGP Monitoring Tool

The Event Manager graphically displays the output of the Response Module. The manager generates event logs for history, audits and debugging. It also provides an interface for configuring the alarm thresholds, contact lists, and other actions.

4 Database Structure

4.1 Use of Internet Routing Registries as a Database for ERIDS

We had planned to use the information in the IRR as authoritative for creating the event database. Our original approach was based on the belief that the IRR was populated with NLRI data and AS routing policies. Due the voluntary nature of the IRR effort, we understood that not all organizations register their routes in the databases making these databases inconsistent and out of date. Our initial assessment turned out to be too optimistic. The NLRI data in the IRR is mostly complete; however, the AS policy data has large gaps. We needed some way to fill in those gaps, or we will have no ability to verify the validity of route. As a result of our research efforts we determined that we need to choose a different approach for creating the event database for ERIDS since the Routing Policy Information stored in the Internet Routing Registry database is imcomplete.

4.2 Detailed Analysis of IRR data

We collected data from a number of sources to analyze the completeness of the IRR. We obtained copies of the several IRR databases from Merit. We retrieved copies of the state of the Route Arbiters at MAE-East and MAE-West for a period of a week from IPMA (the Internet Performance Measurement and Analysis project), which contain the routes intended to be used by all providers at those exchange points. Finally, we extracted routing information from a trace of BGP traffic captured at MAE-East in January, giving us some undigested route information.

We first created a summary of the AS policy information available, by comparing the list of ASes from the route arbiters against the IRR. As of late September, there were 3495 AS's listed in the IRR. There were 1198 AS's appearing in BGP announcements received at MAE-East and MAE-West's route arbiters which were not registered in the IRR. We then measured the completeness of the IRR by using a simplified, static variant of our planned analysis of routing paths. A route announcement's AS PATH attribute is a list of all the ASes a packet must pass through to follow that route.

We can presume that we can check each AS's routing policy by comparing it and its neighbors in an AS PATH against the policy information in the IRR. The IRR's AS-in and AS-out fields describe this information, with AS-in representing the route acceptance policy for an AS and AS-out representing the route distribution policy. We took every route that we retrieved from the route arbiters for MAE-East and MAE-West and considered them as pairs of ASes to check them against the AS-in fields registered in the IRR.

Of all pairs of ASes appearing in these route announcements received by the route arbiters, less than 40% agreed with the record in the IRR that one AS would accept routes from another AS. Of 191262 such pairs, 72525 matched records in the IRR databases, a hit rate of 38%. The percentage rises to 64% (121817 hits) if five (5) AS's are assumed to have perfect policy information (GTE Internetworking, AT&T, IBM Global, internetMCI and Sprint). The result bodes well for future growth of the IRR, and of the ERIDS project's ability to get most of the data it requires from the IRR some time in the future.

4.3 New Approach

Based on these findings We propose to discover by examination and supplement the data missing from the IRR by gathering sets of BGP messages between AS peers (should this be "between BGP speakers"??). We will use methods of statistical confidence (number of samples collected and the values of the samples) to determine the number of messages to capture and whether the data extracted from the messages is of high confidence. If the data extracted is of high confidence, it will be used to augment the data obtained from the IRR. If the data is not of high confidence, appropriate actions will be taken to either increase its confidence or mark it as suspect.

Positive aspects: - Allows the use of existing IRR data - Offers migration path to using only IRR data as the registry compliance increases - Uses current BGP data as basis for "ground truth" - Allows the existing IRR data to be verified against the working BGP derived data (allowing the flagging of stale IRR data)

Residual Issues

The problem inherent in using a history database approach is that it is difficult to detect anomalous behavior when there is nothing other than that very behavior to compare the anomalies against. Much care will be required in choosing the data to collect and the trust we place in that data.

5 Data Processing

6 System Performance

6.1 Detection Accuracy

6.2 Volume of Data

The amount of data transported from the probes to the Intrusion Detection Engine depends on the number of UPDATEs seen at a particular monitoring point. At a populated Network Access Point such as MAE-EAST, where many National Backbones and first tier ISPs peer with each other, the volume of BGP UPDATEs is definitely high. BGP-4 speakers send UPDATE messages whenever a route flaps. Route flapping occurs whenever a BGP speaker sends an UPDATE message to change the status of a prefix due to reachability problems (link problems, mis-configurations, etc.). Routers also send UPDATEs to announce newly formed routes to new prefixes.

The header of a BGP message is 19 octets long. The payload size is variable and depends on the number of prefixes in the Network Layer Reachability Information field, the number of path attributes and the number of withdrawn routes in case the UPDATE is withdrawing routes. BGP messages are encapsulated in TCP segments (20 octets) and in IP datagrams (20 octets). Assuming average BGP UPDATE payloads of 40 octets, the average UPDATE generates around 100 octets of data.

Statistics generated from data collected from route servers at major NAPs by ISI during the period of March-August 1997 indicate that over this period, combined prefix UPDATEs (including both route announcements and withdrawals) peaked at 47.5 UPDATEs/sec with an average of 18 UPDATEs/sec. Using the average number of UPDATEs/sec and an average UPDATE size of 100 octets roughly yields 160 Million octets per day. A hard disk with four Giga-octets of storage capacity could keep up with this volume of data for 20 days or so. It would seem that the Intrusion Probes should be capable of handling and storing this volume of data.

However, the problem of link consumption for the transmission of the data back to the Intrusion Detection Engine still remains. Assuming 160 Million octets worth of data and ignoring transmission and round trip delays, it would take approximately 15 minutes to download a days worth of data from one probe to the Intrusion Detection Engine at T1 speeds (1.544 Mbits/sec). Multiply this number by the number of probes deployed in the network (assume one at each NAP and FIX for a total of 12) and one sees how it would take more than 3.0 hours to download all the data from all the 12 probes every day. This is a non-trivial amount of network traffic and bandwidth consumption.

7 Security Considerations

Acknowledgments

The authors thank Steve Kent and John Zao for their participation in several requirements discussions. Our gratitude to Charlie Lynn, Matt Fredette, Alden Jackson, Dave Mankins, Marla Shepard and Pam Helinek for their contributions to this work. We thank Mary Hendrix and Craig Partridge for reviewing this manuscript.

References

[1] J.C. Mogul, R.F. Rashid, and M.J. Accetta. "The packet filter: An efficient mechanism for user level network code". Technical Report 87.2, Digital WRL, 1987.

[2] S. Kent, R. Atkinson, "Security Architecture for the Internet Protocol", RFC2401, November 1998.

[3] D. Piper, "The Internet IP Security Domain of Interpretation for ISAKMP", RFC2407, November 1998.

[4] M. Condell, C. Lynn, J. Zao "Security Policy Specification Language", Internet Draft draft-ietf-ipsec-spsl-00.txt, November 1998

[5] T.V. Lakshman and D. Stiliadis. "High Speed Policy-based Packet Forwarding Using Efficient Multi-dimensional Range Matching". *Proceedings of ACM Sigcomm 98.*, Sept 1998

[6] H. Edelsbrunner, L.J. Guibas and J. Stolfi. "Optimal point location in a monotone subdivision". *SIAM Journal on Computing*, 15:317-340, 1986.

[7] V. Srinivasan, G. Varghese, S. Suri, W. Waldvogel. "Fast and Scalable Layer Four Switching". *Proceedings of ACM Sigcomm 98.*, Sept 1998.

[8] L. Sanchez, M. Condell "Security Policy System", Internet Draft draft-ietf-ipsec-sps-00.txt, November 1998

[9] S. Kent, "Security Association Map", Private Communication, August 1997.

ERIDS Design Document

Defense Advanced Research Projects Agency
Information Technology Office
External Routing Intrusion Detection System
DARPA Order No. G403/00
Contract No. F30602-98-C-0242

Prepared by: BBN Technologies
10 Moulton Street
Cambridge, MA 02138

For: Air Force Research Laboratory
Information Directorate
26 Electronic Parkway
Rome, NY 13441-4514

```
Outline
  Collection
  Filtering
  Flap Reduction
  Message Format
  Transmission
  Database
    Schema
    Construction
      Authoritative DB
      History DB
    Stability
    Data Structure
  Analysis
  Event Generation
```

o Collection

We start by listening to BGP traffic on the wire. We use the libpcap library to listen to all BGP traffic (TCP port 179) and send those packets to user space.

At program startup, we initialize the libpcap library with a call to pcap_open_live(). We use pcap_compile() to specify a description of the BGP traffic to monitor, and pcap_setfilter() to put it into use. The libpcap library sends data to the program through calls to pcap_dispatch() in the returned pcap_pkthdr struct and a flat u_char array of packet payload data.

The probe thread is event-driven on pcap_dispatch(). We will *compress and *transmit each packet or group of packets through a callback routine passed to pcap_dispatch().

The BPF code to select all traffic to a specific port is included in the paper on BPF (McCanne, Jacobson, Winter 1993 USENIX). It will return entire ethernet packets matching IP/TCP/BGP in the flat u_char array, and a timestamp and length in the pcap_pkthdr structure.

o Filtering

Now that the BGP packets are in memory, we will verify that they are in properly-formatted TCP packets and remove extraneous messages and fields from the messages to save bandwidth. We save the message type, source MAC and IP addresses, and the NLRI, AS_PATH and Next Hop information.

We scan the (flat u_char array) TCP packet for anomalies. We distill down the AS_PATH information, collapsing all AS_SEQUENCE records into one record and all AS_PATH records into one record. We then copy information from the packet header and body into a newly allocated buffer.

o Message Format

The filtered messages will be sent to the IDEs in a TLV-based message format shown below. A well-formed message must consist of a timestamp, a BGP type, a MAC address, a Source address, and may contain zero or more Withdrawn Routes, NLRI, AS Path and Next Hop fields.

```
 0                   1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|          LENGTH               |          TLV sequences        |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                              . . .                            |
```

A TLV sequence consists of 1 byte of Type, 0 or 1 bytes of Length, and a variable number of bytes of Value. Those Types which have well-defined sizes for the associated data will skip the Length field by setting the high-order bit of the Type field.

```
 0                   1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|0|   TYPE      |    LENGTH     |           VALUE
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
or
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|1|   TYPE      |              VALUE
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

The following TLV sequences are defined:

1. Timestamp

A Timestamp is a standard Unix time_t field, 4 octets in length, denoting seconds since 00:00 GMT Jan 1 1970 that this message was observed.

TYPE: (01) hex
LENGTH: None, well-understood (flag is 1)
VALUE: 4 octets of time

2. BGP Type

A BGP Type field is the 1 octet BGP Type from the captured message.

TYPE: (02) hex
LENGTH: None, well-understood (flag is 1)
VALUE: 1 octet of BGP Type

3. MAC Address

A MAC address is a 6 octet link-layer address of the sender of the captured message.

TYPE: (03) hex
LENGTH: None, well-understood (flag is 1)
VALUE: 6 octets of MAC address

4. Source IPv4 Address

A Source IPv4 address is the 4 octet IPv4 address of the sender of the captured message.

TYPE: (04) hex
LENGTH: None, well-understood (flag is 1)
VALUE: 4 octets of IPv4 address

5. Source IPv6 Address

A Source IPv6 address is the 16 octet IPv6 address of the sender of the captured message.

TYPE: (05) hex
LENGTH: None, well-understood (flag is 1)
VALUE: 16 octets of IPv6 address

6. Next Hop IPv4 Address

A Next Hop IPv4 address is the 4 octet IPv4 address represented by the Next Hop field in the original captured message.

TYPE: (06) hex
LENGTH: None, well-understood (flag is 1)
VALUE: 4 octets of IPv4 address

7. Next Hop IPv6 Address

A Next Hop IPv6 address is the 16 octet IPv4 address represented by the Next Hop field in the original captured message.

TYPE: (07) hex
LENGTH: None, well-understood (flag is 1)
VALUE: 16 octets of IPv6 address

8. AS Path

An AS Path is a list of AS Path elements, each of which is a TLV tuple. Each element is a group of 2-octet AS numbers, preceded by a type and a length. The two types are: AS_SEQUENCE (01 hex) and AS_SET (02 hex). The length is a one-octet field.

TYPE: (08) hex
LENGTH: As specified
VALUE: Variable length of AS Path elements.

9. Withdrawn IPv4 Routes

Withdrawn Routes is a list of address prefixes (5 octets each, giving a network address and network size).

TYPE: (09) hex
LENGTH: As specified
VALUE: Variable number of 5 octet IPv4 address prefixes.

10. Withdrawn IPv6 Routes

Withdrawn Routes is a list of IPv6 address prefixes (17 octets each, giving a network address and network size).

TYPE: (0A) hex
LENGTH: As specified
VALUE: Variable number of 17 octet IPv6 address prefixes.

11. IPv4 NLRI

Network Level Reachability Information is a list of IPv4 address prefixes (5 octets each).

TYPE: (0B) hex
LENGTH: As specified
VALUE: Variable number of 5 octet IPv4 address prefixes.

12. IPv6 NLRI

Network Level Reachability Information is a list of IPv6 address prefixes (17 octets each).

TYPE: (0C) hex
LENGTH: As specified
VALUE: Variable number of 17 octet IPv6 address prefixes.

o Flap Reduction

We are concerned about the volume of BGP traffic during a full-scale route storm, where many routes are flapping between routers on the local network. We would like to suppress spurious routing announcements while not suppressing or delaying useful information.

We will cache route messages to accomplish this. This cache will store our TLV format BGP messages (which are timestamped) along with a flag indicating whether the message has been transmitted or not. It will be indexed on IP source address and the NLRI field.

When we receive any BGP message, we check the cache for a matching message. If we do not find a message with the same IP source address and an exactly matching NLRI field, we send the message on for transmission and insert it into the cache, marked as having been transmitted. If we do find a matching message in the cache, then we replace the entry in the cache with the current one, marked as not transmitted, and we do not send the message on for transmission. Periodically we scan the cache, checking message timestamps. If a message is older than a threshold, we remove it from the cache and transmit it if it is marked as not having been transmitted.

This will suppress any route flaps with periodicity less than the threshold, and correctly pass on the stable state (no flaps within one threshold) of the route flap after one threshold time.

This will be implemented as a series of chaining hash tables, one for each 30 second interval within the threshold time and one extra. At any time, one will be 'current' and the others will be previous 30-second times. Insertion is always performed on the current hash table. Lookups check each hash table starting with the current one. When an entry is replaced with a newer entry, the old one is removed from the table it is in and added to the current table. Every 30 seconds, the least current table is cleaned, since all the entries in it are at or above the threshold time. All entries are checked for their transmitted flag, and if they have not been transmitted they are sent off for transmission. Once the table is cleaned it is made the new current table.

o Transmission

To get the data between a probe thread and several IDE threads, local or remote, we need to
take the packets from the buffer, put them through a multiplexer and ship the packets on already established secure connections to the IDEs, and pull them out and put them back in the data structure on the other end.

First, we use the buffer as a flat byte stream, since it has been packed into our message format.

Second, our multiplexer sends copies of the data to each of the open IDE connections made to our sshd.

Third, the IDE at the other end of each connection pulls the routing messages from the ssh connection and puts it into a local buffer. We then hand the buffer on to the analysis engine of the IDE.

We have an sshd on the machine and an ssh client in the BCAP. The sshd is used to accept connections from BCAP/IDEs requesting raw BGP data and from UIs requesting processed events. We categorize them on connection time by having the login handshake identify the data stream requested.

The ssh client is used to request raw BGP data from another BCAP/probe.

o Multiplexer

Multiplexer code brought over from PBSM will handle taking each message and sending it out to each probe subscriber, and taking each event and sending it out to each event subscriber.

o Database

We keep a copy of the IRR and of the HDB locally. Each is composed of three parts (NLRI, AS and Next Hop). We store each of these six in a separate hash table, optimized for rapid access.

. Schema

The NLRI table includes: NLRI (key), AS owning that NLRI, and confidence data (time first seen, time last changed, stability score). NLRI is the network address for that route; AS owning is the autonomous system registered as owning (and having the right to announce) that AS.

The AS table is 2-dimensional. It includes: AS (key), AS-related (key), AS-in policy, AS-in stability score, AS-out predicate, and AS-out stability score. AS-in policy is a field storing the RPSL (routing policy specification language) value for the policy for an AS-in relationship between the two ASes. If null, they have no such relationship. AS-out predicate is a true/false value saying whether these two ASes have an AS-out relationship. Stability scores are detailed in the next section; simply, it is a measure of how stable and thus how trustworthy the relationship (AS-in or AS-out) has been.

The Next Hop table includes: Network address (key), MAC address, AS gatewayed, and confidence data (time first seen, time last changed, stability score). AS gatewayed is the AS using that Next Hop for forwarding.

. Building

Authoritative Database

We build the copy of the IRR (ADB) regularly by taking a fresh copy of the text dumps from Merit and loading them into the IDE. We only store the NLRI, AS and Next Hop records; we merge the half-dozen database files and remove duplicates, we count the number of records required, size hash tables to optimal sizes and copy the data into the hash tables.

AS-in is used to refer to a relationship between two ASes where AS A accepts routing information from AS B. AS-out is the converse, where AS A sends routing information to AS B. Conceptually each is a 2-dimensional grid with a checkmark (and policy and confidence information) at each intersection.

We will store both in a single hash table using both AS A and AS B as keys.

For each AS, we read in the AS-in and AS-out lists, and record each AS-in by (AS, AS-related, AS-in policy and each AS-out by (AS, AS-related, AS-out predicate=true). AS and AS-related are both keys; we hash on the combination and verify both each time we check the hash table.

The NLRI and Next Hop tables are single-key, so we hash on network address and prefix.

History Database

We build the history database (HDB) offline with the MDTool, and we can update it online with the BCAP. In both cases, we record data we have seen and recreate NLRI, AS-in, AS-out and Next Hop data as in the ADB, with the addition of some stability information to tell us how reliable this data seems to be.

We can tell an NLRI simply because each route announcement announces one or more. We can extract each address and the AS originating it from the message, and update the confidence data in the history database based on that information.

Next Hops are very similar to NLRIs; each route announcement contains one and it's associated with the AS that most recently announced it. We take the extra step of storing the MAC address for each Next Hop, correlating the MAC and IP addresses for later use in analysis.

We can determine AS-in and AS-out relationships from AS paths because if an AS A forwards a routing message coming from AS B, AS B must be in AS A's AS-in list and AS A must be in AS B's AS-out list. Thus we can create records of AS-ins and AS-outs we have seen. We can further build an AS-in policy statement by concatenating all the NLRIs we've seen using that AS-in stipulation.

We store them and access them in the history database's hash table in the same way as the ADB's hash table, adding stability information: AS-in is (AS, AS-related, AS-in policy, stability score) and AS-out is (AS, AS-related, AS-out predicate=true, stability score).

. Stability

The history database will be keeping stability scores for NLRIs, Next Hops, AS-ins and AS-outs. These are obtained in two ways:

For NLRIs and Next Hops, we keep a score (real, range 0 to 1) which decays to 1 exponentially, and is decreased by a fixed percentage of its current value each time the NLRI or Next Hop changes.

For AS-ins and AS-outs, we record AS-ins and AS-outs seen that day and once a day we decay the score to 1 if we saw it and 0 if we did not.

By 'decay to 1' we mean calculate s' as 1-exp(-s-ct), where t is the time interval, c is a constant and s is our score. 'Decay to 0' is s'=exp(-s-ct).

. Data Structure

The hash table will not change size much without being rebuilt, so we do not have to worry about growing the table.

When we initially create the table, we want to know how large it needs to be, so we will either need to make two passes through the data or build an intermediate data structure.

We will use the 4.4 BSD db library's hash table routines for the in-memory hash tables. There will be six of them: one each for NLRI, AS pairs and Next Hop for the Authoritative and the History databases.

o Analysis

We analyze and score each BGP message as follows:

```
Apply ad hoc filters.
  We invoke RPSL to screen out specific ASes and NLRIs which do not exist
    Configurable RPSL script
    Deny non-routed networks and out-of-range networks and ASes
  We will have a configuration file for what BGP options are allowed by the
    protocol but never implemented; if they are implemented in the future the
    configuration file can be edited.  Example: BGP speaker is not next hop.

Check the NLRI and owning AS against the databases.
  Look up the NLRI in the ADB.
    If the NLRI is in the ADB, then
      Compare the resulting record's AS-owning field against the origin AS
        (first AS in the AS Path) in the BGP message.
      If the ASes agree, then
        assign it a score of 100% and
        skip to the next step.
      If the ASes disagree, then
        assign it a score of 0%,
        send a Policy Violation:NLRI, Priority 1 event, and
        skip to the next step.
    If the NLRI is not in the ADB, then
      Look up the NLRI in the HDB.
        If the NLRI is in the HDB, then
          Compare the resulting record's AS-owning field against the first
            AS in the AS Path in the BGP message.
          If the ASes agree, then
            assign the message a score equal to the stability score in the HDB
              NLRI record, and
            skip to the next step.
          If the ASes disagree, then
            assign the message a score equal to the inverse of the stability
            score in the HDB record, and
          skip to the next step.
        If the NLRI is not in the HDB, then
          assign it a score of 0%,
          send a New Entry:NLRI, Priority 3 event,
          if HDB modification is enabled, create a new HDB entry for the new
            NLRI with a stability score of 0%,
          and skip to the next step.

Check each AS_PATH against the databases.
  Break up the AS_PATH into overlapping pairs of ASes.
  For each AS pair,
    Look up the AS pair in the ADB.
      If both an AS-in policy and the AS-out predicate are set in the ADB,
        Check the AS-in policy in an RPSL interpreter against the NLRI and AS
          Path in the BGP message.
        If the policy matches, then
          skip to the next step.
        If the policy does not match, then
          assign the message a score of 0%,
          send a Policy Violation:AS-in Priority 1 event, and
          skip to the next step.
      If either an AS-in policy or an AS-out predicate is not set in the ADB,
        ???
      If there is no record for that AS pair in the ADB,
        Look up the AS pair in the HDB.
          If the AS pair record is in the HDB, then
            Check the resulting record's AS-in policy field against the first
              AS in the AS Path in the BGP message.
            If the policy agrees, then
              multiply the message's score by the AS pair stability score in
                the HDB AS pair record, and
              skip to the next step.
            If the policy disagrees, then
              multiply the message's score by the AS pair stability score in
                the HDB AS pair record,
              if HDB modification is enabled, add the NLRI of the present
                BGP message to the HDB AS pair AS-in policy field
                (as 'OR <NLRI>'), and
              skip to the next step.
          If the AS pair record is not in the HDB, then
            assign the message a score of 0%,
            send a New Entry:AS-in Priority 2 event,
            create a new HDB entry for the AS pair with a stability score of 0%
              and an AS-in policy of 'NLRI',
            and skip to the next step.

Check the Next Hop addresses and owning AS against the databases.
  Look up the Next Hop in the ADB.
    If the Next Hop is in the ADB, then
      Compare the resulting record's AS-owning field against the speaker's AS
      (last AS in the AS Path), and the record's IP address against the
      speaker's IP address in the BGP message.
    If the ASes agree and the IP addresses agree, then
      Look up the Next Hop in the HDB.
      If the Next Hop is in the HDB, then
        Compare the resulting record's AS-owning field against the
          speaker's AS (last AS in the AS Path), the record's IP address
          against the speaker's IP address, and the record's MAC address
          against the speaker's MAC address.
        If the ASes agree and the addresses agree, then
          skip to the next step.
        If the MAC addresses disagree, then
          send a Address Change:Next Hop Priority 1 event,
          and skip to the next step.
    If the ASes disagree or the addresses disagree, then
      assign the message a score of 0%,
      send a Policy Violation:Next Hop Priority 0 event, and
      skip to the next step.
  If the Next Hop is not in the ADB, then
    Look up the Next Hop in the HDB.
      If the Next Hop is in the HDB, then
        Compare the resulting record's AS-owning field against the
          speaker's AS (last AS in the AS Path), the record's IP address
          against the speaker's IP address, and the record's MAC address
          against the speaker's MAC address.
        If the ASes agree and the adddresses agree, then
          assign the message a score equal to the stability score in the HDB
            Next Hop record, and
          skip to the next step.
        If the ASes disagree or the addresses disagree, then
          assign the message a score equal to the inverse of the stability
            score in the HDB record,
          send an Untrustworthy:Next Hop Priority 1 event, and
          skip to the next step.
      If the Next Hop is not in the HDB, then
        assign it a score of 0%,
        send a New Entry:Next Hop Priority 0 event,
        if HDB modification is enabled, create a new HDB entry for the new
          Next Hop with a stability score of 0%,
        and skip to the next step.
```

Tally up the score and pass to the event generator.

o Event generation

We now have BGP packets with associated scores. We now filter out all packets with sufficiently high scores, and categorize the remainder with what factor(s) caused the low score. Then, we convert the relevant portions of the packets to text and ship them to a multiplexer, which forwards the events to any UIs requesting events. In text form, the UI filters the events based on message fields deemed relevant by the user.

```
Each event will have the following text fields:
Event Type, Event Subtype will identify the category of event
  (New Entry, Policy Violation, etc) and the subcategory.
"Event!  Type: New Entry; Subtype: NLRI;"
Priority will give an estimate of importance.  Number from 0 to 3, 0 important.
"Priority: 3;"
Relevant Field will give the erroneous field.
"Relevant Field: NLRI;"
Score will give the score for the packet.
"Score: 0%;"
Details will give a human-readable but well-defined description of the
  problem and possible solutions.
"Details: This network has not been seen before.  Check with the owner
  to see if it should be published.;"
Message Summary will summarize the information in the packet.
"Timestamp <value in decimal>, Source IP <dotted octet>, Source MAC
  <coloned hex octet>, NLRIs <series of network addresses>, AS Path
  <series of AS values>, Next Hop <dotted octet>."

The following is the list of events:

New Entry (New NLRI, New AS-in, New AS-out, New Next Hop)
  These are registered when a data element was not found in the databases.
This occurs if such a routing message references an object which is not in the
ADB and has not been seen before.  This may or may not be alarming,
and so is filterable.
  Type: New Entry
  Subtype: NLRI, AS-in, AS-out, Next Hop
  Relevant Field: NLRI, AS Pair or Next Hop
  Score: low
  Details: This <subtype> has not been seen before.  Please check it.

Policy Violation (NLRI Policy Violation, AS-in Policy Violation, Next Hop
     Policy Violation)
  If a field in the routing message disagrees with the information in the ADB,
it's a policy violation.  This is the most clear-cut error we can generate.
(Note: we can't have an AS-out Policy Violation, since we don't have a
trustworthy AS-out denial facility.)
  Type: Policy Violation
  Subtype: NLRI, AS-in, Next Hop
  Relevant Field: NLRI, AS Pair or Next Hop
  Score: 0%
  Details: The <field which tripped the Policy Violation> is against the
policy for this <relevant field>.

Untrustworthy (NLRI Untrustworthy, AS-in Untrustworthy, AS-out Untrustworthy,
    Next Hop Untrustworthy)
  If our analysis and scoring algorithm, operating on the routing message and
the history database, produce a score below our trustworthiness watermark,
we return one or more Untrustworthy messages, identifying the field(s)
```

found untrustworthy and the score(s).
  Type: Untrustworthy
  Subtype: NLRI, AS-in, AS-out, Next Hop
  Relevant Field: NLRI, AS Pair or Next Hop
  Score: score of packet
  Details: The <relevant field> in this packet did not sufficiently agree
with past behavior.

Address Change (Next Hop Address Change)
  If the MAC address in the BGP message doesn't agree with the MAC address
in the HDB, we produce this event. The suspicious address change indicates
either a hardware change (ethernet card replaced) or spoofing of routing
updates.
  Type: Address Change
  Subtype: Next Hop
  Relevant Field: MAC Address
  Score: 0%
  Details: The BGP speaker's MAC address has changed. Please make sure
the speaker is not being spoofed.

BGP Monitoring Tool Design Document

Defense Advanced Research Projects Agency
Information Technology Office
External Routing Intrusion Detection System
DARPA Order No. G403/00
Contract No. F30602-98-C-0242

Prepared by: BBN Technologies
10 Moulton Street
Cambridge, MA 02138

For: Air Force Research Laboratory
Information Directorate
26 Electronic Parkway
Rome, NY 13441-4514

Outline

++

1.0 Introduction:

This document describes the design, implementation, and operation of the BGP Monitoring Tool (BMT). The BMT is a display front end for the BCAD system. The BCAD generates event messages based on BGP traffic received and the existing filtering rules database. These event messages are {logged locally and also} send to all connected BMTs. The BMT has the ability to connect to multiple BCADs simultaneously and display all received messages in a common display window. Various filters can be applied to the received event messages to allow the operator to focus on specific aspects of the received information. The resulting filtered data is available for logging by the BMT.

2.0 Implementation:

This section details the tools and components used to design and implement the BGP Monitoring Tool.

2.1 Development Environment:

The BMT is written in C and compiled using Gnu's gcc (version 2.7.2.3). X-windows is the required windowing system for BMT. The display (windowing) graphics are generated using the Gimp Tool Kit (gtk version 1.0.6). Debugging will be performed with Gnu's gdb (version 4.17). The BMT is designed to be able to run on any Unix implementation that supports Berkley sockets. The application may require compilation on the desired platform before being able to be run.

2.2 Modes of Operation:

The BMT will have two modes of operation: on line and off line. The on line mode of operation describes the real time event message collection performed by the BMT. The BMT receives and logs all event messages transmitted by the BCADs to which it is subscribed. The log file created during on line operation is an unfiltered record of all event messages received by the BMT. The incomming event messages will be subject to filtering by the BMT. The filtered results will be displayed in the BMT's event window. The contents of the BMT's event window will NOT be able to be saved during "on line" operation.

The off line mode allows for the processing of event message log files. The result of the off line processing is displayed in the BMT's event window. The contents of the BMT's event window can be saved in the standard log file format (described in section 2.3).

2.3 Message Format:

The mesage format of the messages passed from BCAD to BMT is defined in the ERIDs design document (See: ERIDS CVS repository for: doc/design.txt). The message contents include:

```
[Message Type]
[Message Sub-Type]
[Priority]
[Score]
[<IDE TLV MESSAGE>]
```

Consider the TLV message a place holder. This format should most likely be a binary format. The TLV contains:

```
[Time stamp]
[Source IP Address]
[Source MAC Address]
[Message type & sub-type]
[NLRIs]
[AS Path]
```

2.4 Log file Format:

There is a set log file format that includes a variable length header followed by a variable length list of event messages. This is a binary format. The log file header format is as follows:

```
[ File id ]           - 4 bytes
[ Version# ]           - 2 bytes
[ Header Length ]  - 2 bytes
[ Filter List ]  - ?? (See Header Length)
[ Separator String ]  - 80 bytes ("#########")
[ Message List ]  - Remainder of file
  [ Message ]     - variable
    [ Message Catagory ]   - 1 byte (Catagory == Event)
    [ Total Length ]     - 2 bytes
    [ Message Type ]     - 1 byte
    [ Message Sub-type ]   - 1 byte
    [ Priority ]     - 1 byte
    [ Score ]     - 1 byte
    [ Relevent Field ]     - variable
      [ Field Type ]       - 1 byte
      [ Field Data ]       - See (Field Type)
    [ Message Summary ]     - variable
      [ Summary Flag ]       - 1 byte
      [ Total Length ]       - 2 bytes
      [ Summary List ]       - variable
[ TLV MESSAGE RECEIVED FROM IDE ]
   .
   .
   .
< EOF >
```

2.5 Message Database File:

This file contains the verbose descriptions of event messages and associated comments. The descriptions and comments are indexed by the event message type and sub-type (hex values). The index values match the values of the messages passed to the BMT by the BCADs. The file format is as follows:

```
[ File id ]           - 4 bytes
[ Version# ]           - 2 bytes    (major, minor)
[ Message Entry ]         - variable
  [ Message Entry Flag ] - 1 byte     (Identifies Entry type [Event, Info,...])
  [ Message Entry Size ] - 2 bytes
  [ Message Type ]        - 1 byte
  [ Message Sub-type ]  - 1 byte
  [ Description Size ]  - 2 bytes
  [ Description ]  - See (Description Size)
  [ Comment Size ]  - 2 bytes
  [ Comment ]  - See (Comment Size)
 .
 .
< Repeat for All Message Entries >
  .
  .
  .
<EOF>
```

2.6 Configuration File:

This file contains the configuration options for the BMT. All filter, logging, database selection, and display color options are retained here. The file is in ascii format. It is to be organized in a keyword:value format. ie: "Event_Dbase_Filename event.dbase"

This first 2 items of the configuration file are require to be the configuration file id (a pre-determined magic number) and the BMT version #. Items that can appear in the configuration file are:

```
Event_Dbase_Filename text string
Help_Filename text string
Log_Filename text string

Log_File_Size int
```

```
Log_File_Number_Per_Day int

Time_Range From (u_long seconds, u_long u-seconds)
Time_Range To (u_long seconds, u_long u-seconds)

Message_Priority Color (R(0-255), G(0-255), B(0-255))

Filter Message_Type int
Filter Message_SubType int
Filter Address text_string
Filter AS int
Filter Message_Priority int
Filter Score (from (int), to (int))

Alias (name (text_string), type (int),
 address(text_string) )

File Format:

[ File id ]
[ Version# ]
################################################################################ (80)
<keyword:value>
.
# Comment
.
<keyword:value>

<EOF>
```

2.7 Connectivity:

One of the salient features of the BMT is its method of connection to the BCAD(s). The list of BCADs to which the BMT is connected determined by the user. The Secure Shell (ssh) application will be integrated into the BMT to enable secure/confidential connections to BCADs. Message passing from BCAD to BMT will be performed using the UDP protocol. Reliability of message delivery to BMTs will be guaranteed by forwarding the UDP message stream through the established ssh connection. Since ssh is a TCP based application, the BCAD UDP message stream gains the reliability of the TCP (ssh) connection.

3.0 Application Display:
========================

The BGP Monitoring Tool follows a model that creates a window header and menu/tool bar that controls many "sibling" windows. The parent (menu/toolbar window) enables the creation or opening of multiple log files. These files can have several sibling windows associated with them. The parent window's commands will be applied to which ever window is considered to be at the top of the displayed log file windows. This top log file window is said to have the "focus". Dialogs or sub-windows that are opened with respect to a log file window will be associated with the parent log file window via file name in the window header.

4.0 Menus:

==========

This section describes the functions available through the pulldown menus. (See end of section for visual representation of menus.)

4.1 File:

The functions under this menu item describe operations available for creating, saving, and printing a log file. The content of this log file is described by the VISIBLE contents of the BMT scrolled message window. The File menu item also supports the About command, which displays an informational dialog about the application. The application can be terminated by selection of the Exit command under this menu item.

4.2 Edit:

The functions under this menu item describe operations that can be performed on the scrolled message window(i.e. Cut, Copy, Paste, Clear). The Find command wil generate a dialog box. This Find dialog box will allow the entry of a search string to be compared with the contents of the scrolled message window. This dialog box will also allow searches on addresses, time stamps, message types, and message priorities.

4.2.1 Find:

The find menu item will call up a dialog box that displays a text entry which the user can enter a search string. The find function applies to the messages displayed in the BMT event window that has the focus. The search will always initiate from the beginning of the BMT event window's buffer and advance to the end. The "Find Next" button will initiate and continue the search. The "Close" button will dismiss the dialog box.

4.3 Options:

This menu item allows access to the filter functions and the BMT configuration funcation.

4.3.1 Filters

This menu command generates a "index card" style of dialog window. At the bottom of the window are two select buttons ("Apply" and "Close"). Selecting the "Apply" button will apply all of the current selections on all of the index cards in this window. Selecting the "Close" button will close the dialog window without enacting any changes. The filtered indexed items are: BCAD, Event Message Type, Time, and Event Message Priority.

4.3.1.1 BCAD
The BCAD filter index card allows for the logging of selected BCADs. The BCAD filter has a scrolled window that lists the BCADs that are being passed through the BMT's event filter. The scrolled filter lists existing BCADs by IP address and alias (if listed). New BCADs can be added or existing BCADs can be removed by using the "Add" and "Remove" buttons at the base of the BCAD filter index card. can be identified by address or alias.

4.3.1.2 Event Messages
The event message filter index card allows for the logging of specific event messages. Selecting a particular event message type and subtype will call up a dialog box specific to the message type. The new dialog box will enable the user to enter a list of information specific to the event message type, sub-type pair. (i.e. Selecting Event Message New Entry : NLRI will call up a dialog box enabling the input of IP addresses and/or IP address ranges).

4.3.1.3 Time
The time filter index card enables filtering by start time and or stop time. The start and stop time entries can be enabled or disabled by the use of their associated toggle buttons. The start time is listed as "From" and the stop time is listed as "To".

4.3.1.4 Priority
The priority filter index allows for the filtering out of event messages based on event message priority. Each priority level has an associated toggle button to enable/disable the logging of that priority.

4.3.1.5 Score
The score filter index allows for the filtering out of event messages based on event message score. The filter will enable message filtering by user defined ranges of score. Filtering an individual value of score can be performed by defining a range of a single score value.

4.3.2 Configuration
This menu command generates a "index card" style of dialog window. At the bottom of the window are four select buttons ("Load", "Apply", "Save", and "Close"). The "Load" and "Save" buttons open a file selection dialog box. This will enable the loading and saving of the current BMT configuration. Selecting the "Apply" button will apply

all of the current selections on all of the index cards in this window. Selecting the "Close" button will close the dialog window without enacting any changes. The configuration indexed items are: Aliases, Display, Message Database, and Logging.

4.3.2.1 Aliases

The Aliases index card enables the creation of aliases for BCAD identification. The BCAD descriptor (IP address, Fully Qualified Domain Name) is added with the BCAD alias. Aliases-Address pairs can be added and/or removed by using the "Add" or "Remove" buttons at the base of the Aliases index card.

4.3.2.2 Display

The Display configuration card allows for the assignment of available pallette colors to each message priority level.

4.3.2.3 Event Message Dbase

The Event Message Database (Dbase) index card can be used to select an event message database for use with the BMT. The index card displays a directory listing box, a file listing box for the current directory, and a path descriptor box. This grouping is a common File:Open or Save dialog format.

4.3.2.4 Logging

The Logging index card enables the specification of log file roll over boundaries. The constraints for the log files will be file size or time. The time boundary will be determined using a combination of a pop up menu listing the number of roll overs per day. Selecting the number of log file roll overs will populate a scrolled list of roll over times. This list of roll over time values will be editable.

4.4 Connect:

This menu item enables the establishment of connections (via ssh) between the BMT and BCAD applications through the "Connect" command. When this menu is selected, the lower half of the menu will display a list of currently connected BCADs. Selecting the BCAD menu item will call up the BCAD connection dialog box. The connection dialog box contains a scrolled list of currently connected BCADs. BCADs can be specified by IP address or Fully Qualified Domain Name. Any passwords or pass phrases required by the "Connect" command will be prompted for by the generation of a dialog box. Connection failures due to authorization failure will be indicated by generation of a dialog box.

4.5 Windows:

This menu item will list all currently open BMT windows. Selecting one of the windows listed in this menu will bring that window to the foreground.

4.6 Help:

This menu will enable some help facility to be displayed. There may also be an "About" function which will briefly display the BMT credits.

4.6.1 Help File:
Currently there is no help file format defined. If there is sufficient time/budget/interest, a ASCII file containing help text should be created for use by the BMT user. The help file should be editable via a text editor.

5.0 Data Structures:
====================
This section describes the data structure heirarchy for the ERIDS:BMT. The structures are written in C so that they may be more easily applied to the GTK GUI design framework.

5.1 Major Structures:

```
#ifndef OBJECTS_H
#define OBJECTS_H

/*
 * Typedefs
 */
typedef struct _BMT_Object BMT_Object;

typedef struct _Event_File_Obj Event_File_Obj;
typedef struct _Event_Display Event_Display;
typedef struct _Event_List_Entry Event_List_Entry;
typedef struct _Connection_Object Connection_Object;

/*
 * Configuration dialog related
 */
typedef struct _Configuration Configuration;
typedef struct _Alias_Dialog Alias_Dialog;
typedef struct _Display_Dialog Display_Dialog;
typedef struct _Logging_Dialog Logging_Dialog;
typedef struct _Event_Dbase_Dialog Event_Dbase_Dialog;

/*
 * Filter dialog related
 */
typedef struct _Filter Filter;
typedef struct _BCAD_Dialog BCAD_Dialog;
typedef struct _Time_Dialog Time_Dialog;
typedef struct _Message_Dialog Message_Dialog;
typedef struct _Priority_Dialog Priority_Dialog;
typedef struct _Score_Dialog Score_Dialog;
```

```
/*
 * Supporting dialogs
 */
typedef struct _Find_Settings Find_Settings;
typedef struct _Challenge Challenge;
typedef struct _Log Log;
typedef struct _Event_Dbase Event_Dbase;
typedef struct _Help_Dbase Help_Dbase;

/*
 * General purpose support structures
 */
typedef struct _FileSelection FileSelection;
typedef struct _CList_Header CList_Header;
typedef struct _TLV TLV;
typedef struct _Message_Summary Message_Summary;
typedef struct _Range Range;
typedef struct _Address_Range Address_Range;
typedef struct _Time_Range Time_Range;
typedef struct _Priority Priority;
typedef struct _IPAddress IPAddress;

/*
 * Linked lists
 */
typedef struct _Connection_List_Entry Connection_List_Entry;
typedef struct _Alias_List_Entry BCAD;
typedef struct _Alias_List_Entry Alias_List_Entry;
typedef struct _Message_Type_List_Entry Message_Type_List_Entry;
typedef struct _Message_Subtype_List_Entry Message_Subtype_List_Entry;
typedef struct _Address_Data_Entry Address_Data_Entry;
typedef struct _AS_Data AS_Data_Entry;

/*
 * nmemonics
 */
typedef u_int   AS;
typedef u_int   MessageType;
typedef u_int   MessageSubType;
typedef u_char  MessagePriority;
typedef u_char  MessageScore;
typedef char    MessageSummary;
typedef u_char  Instance;
typedef struct timeval TimeStamp;

/*
 * Structure definitions:
 * ----------------------
 */
struct _CList_Header {
    char *title;
    int  column_width;
};

/*
 * Log, Event_Dbase, & Help_Dbase defined as individual types
 *  for possible expansion of structure scope
 */
struct _Log
{
    char *filename[LABEL_LENGTH];
    FILE *fp;
};

struct _Event_Dbase
{
    char *filename[LABEL_LENGTH];
    FILE *fp;
};

struct _Help_Dbase
{
    char *filename[LABEL_LENGTH];
    FILE *fp;
};



/*
 * Supporting Structures:
 */
struct _TLV
{
    u_long msg_length;
    u_char *msg;
};

struct _Message_Summary
{
    u_long msg_length;
    char   *msg;
};

struct _IPAddress {
    int    type;        /* IPv4, IPv6 */
    u_char addr[IN6ADDRSZ];
};

struct _Range
{
    int start;
    int stop;
```

```
};

struct _Address_Range {
  IPAddress start;
  IPAddress stop;
};

struct _Time_Range {
  TimeStamp start;
  TimeStamp stop;
};

struct _Priority {
    u_char   enabled;
    u_char   color[3];  /* RED=0, GREEN=1, BLUE=2 */
};

struct _Find_Settings {
  GtkWidget *widget;
  int row;
  char search_string[BUFFER_LENGTH];
  GtkWidget *entry;
  GtkWidget *status_label;
};

struct _Challenge {
    char userid[LABEL_LENGTH];
    char passphrase[LABEL_LENGTH];

    GtkWidget *userid_entry;
    GtkWidget *passphrase_entry;
};

struct _FileSelection {
    GtkWidget *frame;

    GtkWidget *dir_list;
    GtkWidget *file_list;
    GtkWidget *selection_entry;
    GtkWidget *selection_text;
    GtkWidget *main_vbox;
    GtkWidget *load_button;
    GtkWidget *help_button;
    GtkWidget *history_pulldown;
    GtkWidget *history_menu;
    GList     *history_list;
    GtkWidget *fileop_dialog;
    GtkWidget *fileop_entry;
    gchar     *fileop_file;
    gpointer   cmpl_state;
    GtkWidget *fileop_c_dir;
    GtkWidget *fileop_del_file;
    GtkWidget *fileop_ren_file;

    GtkWidget *button_area;
    GtkWidget *action_area;
};

/*
 * Linked list element definitions
 */
struct _Alias_List_Entry {
    Alias_List_Entry *prev;
    Alias_List_Entry *next;

    char                label[LABEL_LENGTH];
    GtkWidget           *label_alias;

    int                 address_type;
    u_char              address[IN6ADDRSZ];
    GtkWidget           *label_address;
};

struct _Connection_List_Entry {
  Connection_List_Entry *prev;
  Connection_List_Entry *next;

  GtkWidget *widget;
  Alias_List_Entry *list;
};

struct _AS_Data_Entry {
    AS_Data_Entry   *prev;
    AS_Data_Entry   *next;

    GtkWidget       *widget;
    char            label[LABEL_LENGTH];
    u_int           as;
};


struct _Address_Data_Entry {
    Address_Data_Entry *prev;
    Address_Data_Entry *next;

    GtkWidget          *widget;

    int                type;              /* IPv4, IPv6 */
    char        label[LABEL_LENGTH];
    Address_Range      address;
};
```

```
struct _Message_Subtype_List_Entry {
    Message_Subtype_List_Entry *prev;
    Message_Subtype_List_Entry *next;

    GtkWidget                  *widget;
    GtkWidget                  *check;

    int                        type;
    int                        enabled;
    int                        pass_all;
    Address_Data_Entry         *address_list;
    AS_Data_Entry              *as_list;
};

struct _Message_Type_List_Entry {
    Message_Type_List_Entry    *prev;
    Message_Type_List_Entry    *next;

    GtkWidget                  *widget;
    GtkWidget                  *check;
    int                        enabled;

    int                        type;
    int                        pass_all;
    GtkWidget                  *container_submsg_list;
    Message_Subtype_List_Entry *submsg_list;
};

struct _Event_List_Entry {
    Event_List_Entry *prev;
    Event_List_Entry *next;

    GtkWidget       *widget;

    TimeStamp       received;
    BCAD     bcad;
    MessageType     type;
    MessageSubType  subtype;
    u_char     address_type;
    Instance     *instance;
    MessagePriority priority;
    MessageScore    score;

    MessageSummary  *summary;
    TLV     *tlv;
};

/*
 * Configuration Dialog Structure and sub-structures
 */
```

```
struct _Event_Display {
    GtkWidget        *widget;
    GtkWidget        *filename;
    GtkWidget        *event_window;
    Event_List_Entry *event_list;
    int number_of_item;
};


struct _Event_Dbase_Dialog {
    GtkWidget *widget;

    GtkWidget *entry_path;
    char      path[LABEL_LENGTH];
    char      file[LABEL_LENGTH];
    FILE      *eventdb_fp;
};

struct _Logging_Dialog {
    GtkWidget       *widget;

    int              type;
    GtkWidget        *size_enable;
    GtkWidget       *size_spinner;
    u_long     size;

    GtkWidget       *time_enable;
    GtkWidget       *time_spinner;
    int               num_files;
    Time_Range        logtime[MAX_LOGS_PER_DAY];
};

struct _Display_Dialog {
    GtkWidget *widget;
    GtkWidget *scrolled_window;
    GtkWidget *graphic_pallette;
    GtkWidget *graphic_priority[MAX_PRIORITY];
    Priority  priority[MAX_PRIORITY];
};

struct _Alias_Dialog {
    GtkWidget *widget;

    GtkWidget *scrolled_window;

    Alias_List_Entry *list;

    GtkWidget *descriptor;
    GtkWidget *alias;

    GtkWidget *rbutton_FDQN;
    GtkWidget *rbutton_IPv4;
    GtkWidget *rbutton_IPv6;
};

struct _Configuration {
    GtkWidget          *widget;

    char               filename[LABEL_LENGTH];
    FILE               *fp;
    int                loaded;
    Alias_Dialog       alias;
    Display_Dialog     display;
    Logging_Dialog     logging;
    Event_Dbase_Dialog event_dbase;
};

/*
 * Filter Dialog structure and sub-structures
 */
struct _Score_Dialog {
    GtkWidget *widget;

    GtkWidget *rbutton_start;
    GtkWidget *entry_start;

    GtkWidget *rbutton_stop;
    GtkWidget *entry_stop;

    Range     score;
};

struct _Priority_Dialog {
    GtkWidget *widget;

    GtkWidget *button[MAX_PRIORITY];
    Priority  state[MAX_PRIORITY];
};

struct _Message_Dialog {
    GtkWidget               *widget;

    GtkWidget               *scrolled_window;
    Message_Type_List_Entry *list;
};

struct _Time_Dialog {
    GtkWidget *widget;

    u_char    from_enable;
    GtkWidget *from_button;
    GtkWidget *from_entry[4];
    u_char    to_enable;
    GtkWidget *to_button;
    GtkWidget *to_entry[4];

    Range     time;
};

struct _BCAD_Dialog {
    GtkWidget *widget;

    GtkWidget *scrolled_window;

    Alias_List_Entry *list;

    GtkWidget *descriptor;

    GtkWidget *rbutton_FDQN;
    GtkWidget *rbutton_IPv4;
    GtkWidget *rbutton_IPv6;
};

struct _Filter {
    GtkWidget *widget;

    BCAD_Dialog     bcad;
    Time_Dialog     time;
    Message_Dialog  message;
    Priority_Dialog priority;
    Score_Dialog    score;
};

/*
 * Top level structures
 */
struct _Connection_Object {
    GtkWidget             *widget;
    GtkWidget             *scrolled_window;
    GtkWidget             *address;
    Connection_List_Entry *connect_list;
};

struct _Event_File_Obj {
    int                  file_index;

    Connection_Object    connect_obj;

    Event_Display event_display;
    Configuration configuration;
    Filter filters;
    Priority priority[MAX_PRIORITY];
    /* Find Function Scratch Pad */
    Find_Settings find;

    /* Event Message Database */
    Event_Dbase         event_dbase;

    /* Help Database */
    Help_Dbase          help_dbase;

    /* Logfile */
    Log                 log_obj;

    /* Scratch Pad */
    char username[LABEL_LENGTH];
    char password[LABEL_LENGTH];
};

struct _BMT_Object {
    GtkWidget      *widget;
    int            width;
    int            height;

    Configuration  default_config;
    Filter    default_filters;

    Event_File_Obj *efo[MAX_FILES];
    GtkWidget      *window_list_entry;
};

#endif OBJECTS_H



5.3 Defines:
------------
#ifndef DEFINE_H
#define DEFINE_H

/* Defines:
 * --------
 */
/* General Definitions */
#define VERSION_MAJOR    0
#define VERSION_MINOR    5

#define MAX_FILES        10
#define MAX_PRIORITY     10
#define MAX_LOGS_PER_DAY 12

#define LABEL_LENGTH     128
```

```
#define BUFFER_LENGTH    255

#ifndef TRUE
#define TRUE             1
#endif

#ifndef FALSE
#define FALSE            0
#endif

#define INVALID_ROW      -1

#define OK               0
#define FAIL             -1

#define SIZE             0
#define TIME             1
#define RED              0
#define GREEN            1
#define BLUE             2

#define ENTRY            0
#define HOUR             1
#define MINUTE           2
#define SECOND           3

#define DEFAULT_LOGFILE_SIZE (1024*1000)   /* 1 Meg */

/* Message Type List */
#define NEW_ENTRY
#define POLICY_VIOLATION
#define UNTRUSTWORTHY
#define ADDRESS_CHANGE

/* Message subType List */
#define NLRI
#define AS_IN
#define AS_OUT
#define NEXT_HOP

/* List Entries */
#define MESSAGE_TYPE_ENTRY
#define MESSAGE_SUBTYPE_ENTRY
#define BCAD_ENTRY
#define ALIAS_ENTRY
#define ADDRESS_ENTRY
#define AS_ENTRY

/*
 * GUI
 */
```

```
/* General */
#define BASE_WINDOW_WIDTH      600
#define BASE_WINDOW_HEIGHT     400

/* Menus */
#define NUMBER_OF_MENUS        1
#define BMT_MAIN_MENU          0

/* CList table headers */
#define NUMBER_OF_ADDR_ALIAS_COLUMNS 2
#define NUMBER_OF_EVENT_DISPLAY_COLUMNS 5


#endif DEFINE_H
```

5.5 Global Variables:

```
BMT_Object    BMTobj;
Configuration default_config;
Filter        default_filter;
```

6.0 Dialog Layout Descriptors:
==============================
This section contains descriptions of BMT dialog boxes in a object GUI format. The format parallels the C++ base Fresco GUI design tool and the C based GTK GUI design tool.

6.1 Alias Dialog

```
      vertical_box
scrolling_window
  horizontal_box
    vertical_box /* alias box */
      label  /* "Alias" */
      horizontal_separator
      label  /* <First Alias Name> */
      .
      .
      .
      label  /* <Last Alias Name> */

    vertical_box /* address box */
      label  /* "Address" */
      horizontal_separator
      label  /* <First Address> */
      .
      .
      .
      label  /* <Last Address> */
 radio_button  /* FQDN */
 radio_button  /* IPv4 */
radio_button  /* IPv6 */
horizontal_box
  vertical_box
    label  /* "Descriptor" */
    label  /* "Alias" */
  vertical_box
    entry  /* Descriptor */
    entry  /* Alias */
horizontal_box
  select_button  /* "Add" */
  select_button  /* "Remove" */
horizontal_box
  select_button  /* "Clear" (Align right) */
```

6.2 Display Dialog

```
      vertical_box
        horizontal_box
  label     /* "Palette" */
scrolling_window
  graphical_insert  /* Color Palette */
horizontal_box
  label     /* "Message Priority" */
horizontal_box
  label     /* "0" */
  graphical_insert  /* Color Square */
horizontal_box
  label     /* "1" */
  graphical_insert  /* Color Square */
.
.
.
horizontal_box
  label     /* "n" */
  graphical_insert  /* Color Square */
```

6.3 Logging Dialog

```
      vertical_box
        horizontal_box
  label     /* "Rollover Boundry" */
horizontal box
  radio_button     /* size */
  horizontal box
    label     /* "Size" */
    spin button     /* (1.0 - 10.0) Up/Down Arrow rolling type */
horizontal box
  radio button     /* Logs per day */
  horizontal box
    label     /* "Rollovers/Day" */
    spin button     /* (1-12) Up/Down Arrow rolling type */
```

```
6.4 Event Dbase Dialog
      vertical box
        horizontal box
  vertical box
    label      /* "Directory" */
    scrolled_window
      DIRECTORY_ENTRY
  vertical box
    label      /* "Files" */
    scrolled_window
      FILE_ENTRY
label      /* File path */
entry      /* File path */
horizontal box
  select_button      /* "Load" */


6.5 BCAD Dialog
      vertical box
scrolling_window
  horizontal box
    vertical box /* alias box */
      label  /* "Alias" */
      horizontal_separator
      label  /* <First Alias Name> */
      .
      .
      .
      label  /* <Last Alias Name> */

    vertical box /* address box */
      label  /* "Address" */
      horizontal_separator
      label  /* <First Address> */
      .
      .
      .
      label  /* <Last Address> */
radio_button  /* FQDN */
radio_button  /* IPv4 */
radio_button  /* IPv6 */
horizontal box
  label  /* "Descriptor" */
  entry  /* Descriptor */
horizontal box
  select_button  /* "Add" */
  select_button  /* "Remove" */
```

6.6 Time Dialog

```
      vertical box
        horizontal box  /* Start Time */
  toggle_button
  label  /* "From" */
  spin button   /* Hour */
  spin button   /* Minute */
  spin button   /* Second */
        horizontal box  /* Stop Time */
  toggle_button
  label  /* "To" */
  spin button   /* Hour */
  spin button   /* Minute */
  spin button   /* Second */
```

6.7 Message Dialog

```
      scrolled_window
        vertical box /* Template for Message type/sub-type <0>*/
          horizontal box
    toggle_button
    label /* Message type */
  vertical box          /* Message sub-type container */
    horizontal box       /* Message subtype <0> */
      toggle button
      label
      select button      /* Button to call up specific dialog */
    horizontal box /* Message subtype <1> */
      toggle button
      label
      select button      /* Button to call up specific dialog */
      .
      .
      .
    horizontal box /* Message subtype <n> */
      toggle button
      label
      select button      /* Button to call up specific dialog */
        .
        .
        .
          horizontal box /* Template for Message type/sub-type <n>*/
    toggle_button
    label /* Message type */
  horizontal box /* Align Center */
  vertical box /* Message sub-type container */
    horizontal box     /* Message subtype <0> */
      toggle button
      label
      select button      /* Button to call up specific dialog */
    horizontal box /* Message subtype <1> */
```

```
      toggle button
      label
      select button     /* Button to call up specific dialog */
      .
      .
      .
    horizontal box /* Message subtype <n> */
      toggle button
      label
      select button     /* Button to call up specific dialog */
```

```
6.8 Priority Dialog
      vertical box
        label  /* Priority Logged */
horizontal box
  toggle_button
  label  /* "0" */
horizontal box
  toggle_button
  label  /* "1" */
.
.
.
horizontal box
  toggle_button
  label  /* "n" */
```

```
6.9 Score Dialog
      vertical box
        horizontal box  /* Start Score */
  toggle_button
  label  /* "From" */
  spin button  /* % (0 - 100) */
        horizontal box  /* Stop Time */
  toggle_button
  label  /* "To" */
  spin button  /* % (0 - 100) */
```

```
6.10 Find Dialog
      Window   /* Find */
        vertical box
  horizontal box
    label   /* "Find:" */
    text entry   /* for enter search string */
  horizontal separator
  horizontal box
    button       /* "Find" button */
    button       /* "Find Next" button */
    button       /* "Clear" button */
    button         /* "Close" button */
  horizontal separator
  horizontal box
    label          /* "Status: " */
    label          /* for displaying status info */
```

6.11 Password Dialog

```
      horizontal box
        vertical box
  label   /* "Username" */
  label   /* "Password" */
vertical box
  entry   /* username */
  entry   /* password */
```

BCAD Design and Use Document

Defense Advanced Research Projects Agency
Information Technology Office
External Routing Intrusion Detection System
DARPA Order No. G403/00
Contract No. F30602-98-C-0242

Prepared by: BBN Technologies
10 Moulton Street
Cambridge, MA 02138

For: Air Force Research Laboratory
Information Directorate
26 Electronic Parkway
Rome, NY 13441-4514

BCAD Design and Use

This file documents BBN's External Routing Intrusion Detection System (ERIDS), a system for monitoring BGP-4 announcements.

The External Routing Intrusion Detection System (ERIDS).

This manual documents ERIDS, a system for monitoring BGP-4 route announcements. It documents release 0.1 of the system.

Introduction

The Border Gateway Protocol 4 (BGP-4) is a protocol designed to distribute external routing information to the routers of distinct Autonomous Systems in an internetwork (*note References: Rekhter and Li).

Problems with BGP-4

The integrity of the global, external routing infrastructure and the correct operation of routers depend on each router, or "BGP speaker", only having correct forwarding information, and only announcing correct forwarding information to others. Specifically, the correct operation of BGP-4 depends upon the integrity, authenticity, and recentness of the routing information it distributes as well as each BGP speaker's processing, storing, and distribution of this information in accordance with both the BGP specification and the routing policies of the BGP speaker's autonomous system.

BGP-4 is quite vulnerable to a variety of malicious attacks both in theory and in practice, due to the lack of a scalable means to verify the authenticity and legitimacy of the BGP control traffic with which routers update their Routing Information Bases.

For example, the communication links between BGP neighbors are subject to active and passive wiretapping. Fictitious BGP messages can be injected into a link in a "spoofing attack". Intercepted BGP messages can be replayed over a link at a later time, in a "replay attack". Regardless of the specifics of an attack, the acceptance of erroneous BGP UPDATE messages and/or the suppression of UPDATEs may cause route selections that are not consistent with the current state of the internetwork, or that violate local routing policies. In the absence of any attack, the erroneous configuration or malfunction of normally well-behaved BGP speakers can also produce these same problems.

BBN's S-BGP

BBN Technologies is currently developing a system of countermeasures to secure BGP-4 (*note References: Lynn). This system, S-BGP, relies on a public-key infrastructure that follows IP address space allocation and Autonomous System number assignment, and features a new BGP transitive path attribute containing signatures that verify the validity of the routes that a BGP speaker advertises. This system is backwards-compatible with existing BGP-4 implementations and does not require hardware modifications to existing routing gear. However, deployment of the proposed countermeasures is a nontrivial task. It requires deployment of additional hardware and software by Internet Service Providers (ISPs) in support of these countermeasures and establishment and administration of a public key infrastructure by registration authorities and ISPs, all on the scale of the worldwide Internet. It may be several years before any deployment would be completed.

ERIDS

In the meantime, BBN has also developed the External Routing Intrusion Detection System, or ERIDS. ERIDS is a system designed to provide Network Operation Centers (NOCs) with information describing BGP-4 problems due to malicious attacks or misconfigurations. Two requirements for ERIDS were that it must leverage the existing routing infrastructure and must demonstrate backward compatibility with existent networking gear and legacy systems.

There are three main features of the ERIDS system:

The BGP capture feature.
ERIDS devices, installed at shared and private interconnects, will "snoop" on normal, existing BGP communications and feed BGP messages to one or more BGP Intrusion Detection Engines.

The BGP Intrusion Detection Engine.
The Intrusion Detection Engine (IDE) is the ERIDS feature that analyzes BGP messages fed to it by BGP capture devices. The information base used by the IDE includes routing policy information obtainable from Internet routing registries, and a history database that it maintains itself. The IDE generates *events* when it determines that something is wrong at one or more interconnects, and feeds those events to one or more BGP Monitoring Tools.

The BGP Monitoring Tool.
The BGP Monitoring Tool (BMT) is the ERIDS graphical user interface feature. The BMT allows an operator to connect to and display the events from multiple IDEs in a common display window. Various filters can be applied to the received event messages to allow the operator to focus on specific activity, and the resulting filtered data can be written to a log.

The first two ERIDS features are realized in one program, called the BGP Capture and Analysis Daemon (BCAD, or 'bcad'). The BMT feature is realized in a program called 'bmt'.

Miscellaneous
=============

The ERIDS distribution also contains three other pieces of code of note:

'bgpsh'
The BGP shell 'bgpsh' is a command-line driven program that can drive BGP sessions with multiple peers. It performs no true routing function; it merely takes a "script" of BGP messages to perform with the various peers, and performs it.

'bnf-to-parser'
The 'bnf-to-parser' tool is a Perl script that can automatically generate 'lex' and 'yacc' input from a single BNF specification. It is used to generate the scanner and the parser for 'bgpsh'.

'libscap'
The stream capture library 'libscap' is a very simple, almost unfinished library for reconstructing TCP streams from captured packets. It works best when used in conjunction with 'libpcap'.

Building ERIDS
****

To build all of ERIDS, you must have already compiled and installed the following libraries and programs:

'libpcap'
    The 'libpcap' library is used by 'bcad' to capture packets carrying BGP messages. It is available at 'ftp://ftp.ee.lbl.gov/libpcap.tar.Z'.

'gtk+'
    The 'gtk+' library is used by 'bmt' as its graphical user interface library. At least 'gtk+-1.2.0' is required, and it is available at 'ftp://ftp.gtk.org'.

'ssh'
'sshd'
    The 'ssh' program can be used by 'bcad' and 'bmt' to connect to remote 'bcad's and retrieve captured BGP messages or generated IDE events.

    The 'sshd' program can be used by 'bcad' to allow remote 'bcad's and 'bmt's to connect and retrieve captured BGP messages or generated IDE events. *Note that for simplicity and security, a special, ERIDS-only version of 'sshd' is built for this purpose.* This version is called 'erids-sshd'.

    The 'ssh' distribution (which includes 'sshd') is available at 'ftp://ftp.ssh.fi'; version 1.2.26 has been tested with ERIDS.

'ucd-snmp'
    The University of California at Davis SNMP package can be extended to provide ERIDS support. The 'ucd-snmp' distribution is available at 'http://ucd-snmp.ucdavis.edu'; version 3.6.2 has been tested with ERIDS.

'readline'
    The GNU 'readline' and 'history' libraries are purely optional, but if present, allow 'bgpsh' to feature command-line editing and history.

Once these needed libraries and programs have been compiled and installed, begin compiling ERIDS by unpacking the distribution '.tar' file. The distribution includes a 'configure' script that prepares the ERIDS sources to compile on your system. This script can take command-line options that control how the sources will be built:

* '--prefix='DIR

  Normally, 'configure' will prepare the distribution to be installed under '/usr/local/bin', '/usr/local/lib', etc. To make it use an installation prefix other than '/usr/local', give 'configure' the option '--prefix='DIR.

* '--without-x'

If you do not have the X window system, or do not have the 'gtk+' library installed, you will not be able to build the BMT, and you should give the '--without-x' option to 'configure'.

* '--with-sshd-build='DIR

  To build and install the special, ERIDS-only 'erids-sshd', use this 'configure' option, where DIR is the name of the directory containing the 'sshd' sources and objects. The 'bcad' build process will automatically build and install this special version.

* '--without-sshd-build-dir'

  Alternately, if you do not wish to build the special ERIDS-only 'sshd', you can use this 'configure' option.

* '--with-ucd-snmp-source='DIR

  To build (but not install) the special, ERIDS-enabled version of the UCD SNMP package, use this 'configure' option, where DIR is the name of the directory containing the UCD SNMP 'configure' script. The ERIDS configuration and build process will automatically prepare and build that distribution with ERIDS support.

* '--without-sshd-build-dir'

  Alternately, if you do not wish to build the special ERIDS-enabled version of the UCD SNMP package, you can use this 'configure' option.

Once you've decided on any of the above 'configure' options, follow these steps to prepare and build the distribution:

1. 'sh ./configure 'OPTIONS

   This will prepare the distribution for compiling. If you get error messages here, your machine might not be supported by this release. (*Note Bugs: for how to report bugs in the distribution.)

2. 'make'

3. 'make install'

   This will install the ERIDS distribution under the prefix you selected with '--prefix', or it defaults to '/usr/local'.

Notes about 'erids-sshd'

If you built 'erids-sshd', by default it installs into

'/usr/local/sbin' (or under whatever other '--prefix' you specified to the ERIDS 'configure').

However, since 'erids-sshd' is largely built from the same 'sshd' sources and objects you originally compiled, all of its 'sshd'-type behavior will be governed by the same arguments you originally gave to the 'ssh' 'configure'. This means that this 'sshd' will use the same configuration files and host key that your normal 'sshd' uses.

This may seem confusing. You may ask, if a separate 'sshd' really *is* needed, shouldn't it be completely separate, with its own host key and configuration files? We believe the answer is no. 'erids-sshd' is a special, "captive" version of your normal 'sshd', that allows only the 'erids' user to log in and run one very special program, 'bcadsh'. 'bcadsh' is the program that connects to the local 'bcad' and asks for either BGP or IDE information to pass back to the remote client.

Since 'bcadsh' must run as root, to save the user from having to create the 'erids' user and then install 'bcadsh' setuid-root, we decided to create 'sshd-erids'. 'sshd-erids' is your normal 'sshd' linked with the 'bcad-sshd' object; this object captures 'libc' functions to create the illusion of a real 'erids' user, and to force that the 'erids' user only run the 'bcadsh' program.

There is one real component to this 'erids' user - an 'sshd' 'authorized_keys' file. This file lists the RSA identities that are allowed to "log in as" the 'erids' user. This list of keys combined with the 'bcad' '--bgp-server' and '--ide-server' client lists form a complete 'bcad' access-control system. The 'erids' 'authorized_keys' file is expected to be found under '/usr/local/etc/erids/.ssh' (or similarly under any '--prefix' you gave to the ERIDS configure).

Using the BCAD
****

The BGP Capture and Analysis Daemon (BCAD, or 'bcad') is a single program that performs the ERIDS BGP capture and Intrusion Detection Engine functions. It is also able to feed the BGP messages it captures, and the IDE events it generates, to clients. A BGP message client is normally another 'bcad', and an IDE event client is normally a BMT.

The 'bcad' program is normally installed in '/usr/local/sbin'. It may take several command-line options:

'--bgp-probe FILTERFILE INTERFACE'
    This option turns on the BGP capture feature of 'bcad'. INTERFACE is the name of the network interface that 'bcad' should listen on, or the name of a 'tcpdump' capture file.

```
     FILTERFILE is the name of a file containing a `tcpdump'
     expression.  This expression should select the packets belonging
     to the BGP sessions you want `bcad' to monitor.  To capture all
     BGP traffic on the wire, this could be just:

          port 179

     However, normally this will be an expression to capture only the
     BGP traffic between certain selected peers.

`--bgp-client SOURCEFILE'
     This option turns on the BGP client feature of `bcad'.  SOURCEFILE
     is the name of a file listing the *sources* that this `bcad' should
     read BGP information from.  The format of this file is:

        * Blank lines and lines that begin with `#' are ignored.

        * `ssh' HOST [[USERNAME] PORT]

          Each line of this form specifies that `bcad' should use `ssh'
          to connect to HOST to retrieve BGP information.  The optional
          PORT specifies the port the ERIDS sshd is listening on on
          HOST.  The optional USERNAME specifies the username that the
          local `bcad' should use to connect to the remote `bcad'.

          `bcad' will always try to reestablish `ssh' connections that
          have closed; it uses a backoff mechanism to throttle the
          reconnection attempts.

        * `file' FILENAME

          Each line of this form lists a file that `bcad' should read
          BGP information from.

        * `full' COMMAND

          Each line of this form specifies a full command that `bcad'
          should execute to retrieve BGP information.  `bcad' will
          always try to reopen `full' commands that have closed; it
          uses a backoff mechanism to throttle the reopen attempts.

`--bgp-server CLIENTFILE'
     This option turns on the BGP server feature of `bcad'.  CLIENTFILE
     is the name of a file listing the *clients* that are allowed to
     connect to this `bcad' and retrieve captured BGP messages.  The
     format of this file is:

        * Blank lines and lines that begin with `#' are ignored.

        * Any other line is the hostname or IP address of an allowed
          client.
```

'--ide DATABASEDIR'
    This option turns on the Intrusion Detection Engine feature of 'bcad'. DATABASEDIR is the name of a directory containing the compiled databases used to seed the IDE information base. There are two sets of compiled databases: one contains *authoritative* information, and the other contains *history* information. Initially, the history databases are nonexistent. To compile the authoritative databases, run the ERIDS 'dbgen' script with an IRR-style database file on its standard input:

```
          cd DATABASEDIR
          dbgen < IRRDBFILE
```

    'dbgen' always writes its output to the current directory.

'--ide-server CLIENTFILE'
    This option turns on the IDE server feature of 'bcad'. The '--ide' switch *must* be used with this switch. CLIENTFILE is the name of a file listing the *clients* that are allowed to connect to this 'bcad' and retrieve generated IDE events. The format of this file is:

   * Blank lines and lines that begin with '#' are ignored.

   * Any other line is the hostname or IP address of an allowed client.

None of these options preclude any others; for example, it is possible to give 'bcad' the '--bgp-probe', '--bgp-client', and '--bgp-server' switches. The BGP data collected and served will be the data captured by '--bgp-probe' and received through '--bgp-client'.

Normally, 'bcad' puts itself into the background, leaving behind a PID file '/var/run/bcad.pid' (or, if that directory does not exist, '/etc/bcad.pid').

Sending a 'SIGHUP' to 'bcad' causes it to reload. The meaning of a reload is different for the different 'bcad' features:

'--bgp-probe'
    The 'libpcap' filter file is reloaded.

'--bgp-client'
    The sources file is reloaded. Any new sources are opened, and any removed sources are closed.

'--bgp-server'
    The clients file is reloaded. Any new clients are allowed to connect, and any old clients are bumped off.

'--ide'
    The authoritative databases are reloaded.

'--ide-server'
    The clients file is reloaded. Any new clients are allowed to connect, and any old clients are bumped off.

When '--ide' is used, sending a 'SIGFPE' to 'bcad' causes it to dump its history databases to DATABASEDIR.

Using the BMT

This is the little I could cull from existing documents. More content and editing is needed - fredette.

The BGP Monitoring Tool (BMT) is the client portion of the ERIDS system. Then BMT serves as the event manager and message interpretation facility for a network operator. Messages coming from BCADs are decoded and translated into human readable format using a local event interpreter database. This tool allows the operator to establish a connection to one or more BCADs in order to receive intrusion warning or alert messages from public Network Access Points and from direct interconnect sessions simultaneously.

The Event Manager graphically displays the output of the Response Module. The manager generates event logs for history, audits and debugging. It also provides an interface for configuring the alarm thresholds, contact lists, and other actions.

ERIDS on SNMP

ERIDS can also be monitored with the Simple Network Management Protocol (SNMP). The ERIDS distribution includes a module that extends the University of California at Davis (UCD) SNMP agent to present BCAD IDE events in the form of an SNMP table.

The UCD SNMP distribution can be found at 'http://ucd-snmp.ucdavis.edu'. The ERIDS module has been tested with version 3.6.2 of their package. To build this package with the ERIDS support, unpack it and rerun the ERIDS 'configure' script with the '--with-ucd-snmp-source='DIR option, where DIR is the name of the directory containing the UCD 'configure' script. The ERIDS 'configure' will take care of adding the necessary ERIDS sources to the UCD build tree and reconfiguring it, and the ERIDS 'Makefile's will build it.

To use ERIDS features of this modified 'snmpd', first follow the instructions in the normal UCD documentation on how to install, configure, and start a normal 'snmpd'. Once this is successful, add the following *single* line to 'snmpd.conf' to have 'snmpd' become a client of a BCAD running somewhere:

```
     erids store TABLESIZE source ssh -n -l erids -i IDENTITY -a -x
       -p PORT HOST bcadsh ide
```

Replace TABLESIZE with the maximum number of entries the ERIDS event table should hold. Replace HOST and PORT with the hostname and port number where an ERIDS 'sshd' is listening, and replace IDENTITY with the path to the 'ssh' identity file needed to connect to the 'erids' account there.

Once the 'snmpd' has connected to the remote BCAD (remember to put the machine running 'snmpd' in the '--ide-server' clients list of the BCAD), you can walk the ERIDS subtree of the MIB. For example, on the same machine where 'snmpd' is running:

```
     $ snmpwalk -v 1 localhost COMMUNITY \
       .iso.org.dod.internet.private.enterprises.bbn.products.erids
```

will walk the entire current contents of the ERIDS MIB subtree. To view the current events table in a more intuitive form, do:

```
     $ snmptable -v 1 localhost COMMUNITY \
       .iso.org.dod.internet.private.enterprises.bbn.products.erids.eridsEventTable
```

The BGP shell
****

ERIDS also comes with a BGP shell called 'bgpsh'. This is a command-line driven program that drives BGP sessions with multiple peers. You present it with a script describing the sessions to open and a sequence of messages to send on those sessions, with optional delays.

The 'bgpsh' usage is:

```
     $ bgpsh -h
     usage: bgpsh [<options>]
     where <options> are:
       -h, --help                display this usage
       -v, --verbose             be verbose
       -p PORT, --port PORT      listen on port PORT
       -d, --debug               turn on debugging
```

The '--verbose' switch displays messages received from peers. The '--port' switch tells 'bgpsh' to listen on a port different from the normal BGP port. Using '--debug' one or more times produces increasing amounts of debugging information.

Once started, 'bgpsh' presents a simple prompt:

```
$ bgpsh -v
bgpsh>
```

There is no 'help' command. The shell ignores blank lines and lines that begin with '#'. There is a simple variable system. Any line of the form:

```
NAME=VALUE
```

defines the expansion of NAME to be VALUE. Variable substitution is done when '$'NAME is seen on an input line.

A command that has had all variables substituted for, and is not itself a variable definition, has the general form:

```
peer IPADDRESS [wait WAIT-TIME] BASIC-COMMAND
```

Each line begins by specifying the IPADDRESS of the peer it is associated with, and an optional WAIT-TIME in seconds. All of the command lines associated with the same peer are executed in order, and execution of commands to different peers are interleaved. A command is executed WAIT-TIME seconds after the last command to the same peer completes, or WAIT-TIME seconds from the time the shell received the command, whichever is later.

The possibilities for BASIC-COMMAND are:

'open' ['port' REMOTE-PORT] ['passive'] ['bgp-id' LOCAL-BGP-ID]
    'local-as' LOCAL-AS ['hold-time' HOLD-TIME]

> This must be the first command used with a given peer. The optional REMOTE-PORT is the port to connect to the peer on. The optional 'passive' tells 'bgpsh' to wait for the peer to make the connection, without trying to establish it itself. The optional LOCAL-BGP-ID specifies what to claim as the local BGP identifier in the 'OPEN' message. The LOCAL-AS number and optional HOLD-TIME also go into the 'OPEN' message. The default hold time is two minutes.
>
> An 'open' command completes only when the peer session is successfully established.

'notification' NOTIFY-CODE [NOTIFY-SUBCODE]

> This sends a 'NOTIFICATION' message to a peer and closes the session.
>
> NOTIFY-CODE must be one of 'message', 'open', 'update', 'hold-time', 'fsm', 'cease', or an integer.

NOTIFY-SUBCODE, if present, must be one of ‘no-sync’, ‘bad-length’, ‘bad-type’, ‘bad-version’, ‘bad-as’, ‘bad-id’, ‘bad-parameter’, ‘bad-auth’, ‘bad-hold-time’, ‘bad-attributes’, ‘bad-well-known-attribute’, ‘missing-well-known-attribute’, ‘bad-attribute-flags’, ‘bad-attribute-length’, ‘bad-origin’, ‘as-routing-loop’, ‘bad-next-hop’, ‘bad-optional-attribute’, ‘bad-network’, ‘bad-as-path’, or an integer.

‘keepalive’

This sends a ‘KEEPALIVE’ message to a peer. This command is normally not used, since ‘bgpsh’ automatically sends keepalives often enough to respect a peer session's hold time.

‘update’ [‘withdraw’ ‘nlri’ PREFIXES] [‘advertise’ PATH-ATTRIBUTES ‘nlri’ PREFIXES]

This sends an ‘UPDATE’ message to a peer. An ‘update’ command must contain either a ‘withdraw’ clause or an ‘advertise’ clause, or both.

PREFIXES is a comma-separated list of prefixes to advertise or withdraw.

Valid PATH-ATTRIBUTES are:

‘origin’ {‘igp’ | ‘egp’ | ‘incomplete’}

This specifies the origin of the announced prefixes.

‘as-path’ AS-PATH-SEGMENTS

This specifies the AS path of the announced prefixes. A single AS-PATH-SEGMENT is either ‘set’ or ‘sequence’ followed by a list of AS numbers.

‘next-hop’ IPADDRESS

This specifies that the next hop of the announced prefixes should be IPADDRESS.

‘med’ INTEGER

This specifies that the MED of the announced prefixes should be INTEGER.

‘local-pref’ INTEGER

This specifies that the localpref of the announced prefixes should be INTEGER.

‘atomic-aggregate’

This specifies that the announced prefixes are to be considered atomic aggregates.

‘aggregator’ AS-NUMBER IPADDRESS

This specifies that the announced aggregates were generated by IPADDRESS in AS AS-NUMBER.

Bugs
****

The programs in the ERIDS release all have bugs and shortcomings. This section captures the known bugs.

Reporting Bugs
==============

For sure, many more bugs and shortcomings exist. Bug reports should be mailed to <erids-core@cotton.ir.bbn.com>. The best bug reports include:

* Input files and command lines that demonstrate the bug.

* The version of the ERIDS release with the bug.

* A suggested patch (please use 'diff -c') and 'ChangeLog' entry.

Copying
****

References
****

☆U.S. GOVERNMENT PRINTING OFFICE: 2000-610-055-10022

DISTRIBUTION LIST

addresses	number of copies
AFRL/IFG ATTN: ROBERT KAMINSKI 525 BROOKS ROAD ROME, NEW YORK 13441-4505	5
PBN TECHNOLOGIES 10 MOULTON STREET CAMBRIDGE, MA 02138	2
AFRL/IFOIL TECHNICAL LIBRARY 26 ELECTRONIC PKY ROME NY 13441-4514	1
ATTENTION: DTIC-OCC DEFENSE TECHNICAL INFO CENTER 8725 JOHN J. KINGMAN ROAD, STE 0944 FT. BELVOIR, VA 22060-6218	1
DEFENSE ADVANCED RESEARCH PROJECTS AGENCY 3701 NORTH FAIRFAX DRIVE ARLINGTON VA 22203-1714	1
ATTN: NAN PFRIMMER IIT RESEARCH INSTITUTE 201 MILL ST. ROME, NY 13440	1
AFIT ACADEMIC LIBRARY AFIT/LDR, 2950 P.STREET AREA B, BLDG 642 WRIGHT-PATTERSON AFB OH 45433-7765	1
AFRL/MLME 2977 P STREET, STE 6 WRIGHT-PATTERSON AFB OH 45433-7739	1

AFRL/HESC-TDC 2698 G STREET, BLDG 190 WRIGHT-PATTERSON AFB OH 45433-7604	1
ATTN: SMDC IM PL US ARMY SPACE & MISSILE DEF CMD P.O. BOX 1500 HUNTSVILLE AL 35807-3801	1
TECHNICAL LIBRARY D0274(PL-TS) SPAWARSYSCEN 53560 HULL ST. SAN DIEGO CA 92152-5001	1
CDR, US ARMY AVIATION & MISSILE CMD REDSTONE SCIENTIFIC INFORMATION CTR ATTN: AMSAM-RD-OB-R, (DOCUMENTS) REDSTONE ARSENAL AL 35898-5000	2
REPORT LIBRARY MS P364 LOS ALAMOS NATIONAL LABORATORY LOS ALAMOS NM 87545	1
ATTN: DEBORAH HART AVIATION BRANCH SVC 122.10 FOB10A, RM 931 800 INDEPENDENCE AVE, SW WASHINGTON DC 20591	1
AFIWC/MSY 102 HALL BLVD, STE 315 SAN ANTONIO TX 78243-7016	1
ATTN: KAROLA M. YOURISON SOFTWARE ENGINEERING INSTITUTE 4500 FIFTH AVENUE PITTSBURGH PA 15213	1
USAF/AIR FORCE RESEARCH LABORATORY AFRL/VSOSA(LIBRARY-BLDG 1103) 5 WRIGHT DRIVE HANSCOM AFB MA 01731-3004	1

ATTN: EILEEN LADUKE/D460 MITRE CORPORATION 202 BURLINGTON RD BEDFORD MA 01730	1
OUSD(P)/DTSA/DUTD ATTN: PATRICK G. SULLIVAN, JR. 400 ARMY NAVY DRIVE SUITE 300 ARLINGTON VA 22202	1

MISSION
OF
AFRL/INFORMATION DIRECTORATE (IF)

The advancement and application of Information Systems Science and Technology to meet Air Force unique requirements for Information Dominance and its transition to aerospace systems to meet Air Force needs.